AD-A119 041   AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH   F/G 6/16

COMPUTER AUTOMATION OF THE THERMAL PULSE TECHNIQUE FOR LOCAL BL--ETC(U)

1982   K L BAUM

UNCLASSIFIED   AFIT/CI/NR/82-45T   NL

1 of 1

AD A
119041

END
DATE
FILMED
10 82
DTIC

UNCLASS

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/CI/NR/82-45T	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Computer Automation of the Thermal Pulse Technique for Local Blood Flow Measurements		5. TYPE OF REPORT & PERIOD COVERED THESIS/~~DISSERTATION~~/
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Kurt Lewis Baum		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS AFIT STUDENT AT: University o. Illinois		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS AFIT/NR WPAFB OH 45433		12. REPORT DATE 1982
		13. NUMBER OF PAGES 79
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASS
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

APPROVED FOR PUBLIC RELEASE: IAW AFR 190-17

30 AUG 1982

LYNN E. WOLAVER
Dean for Research and
Professional Development
AFIT, Wright-Patterson AFB OH

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

ATTACHED

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASS

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

82 09 07 418

COMPUTER AUTOMATION OF THE THERMAL PULSE TECHNIQUE FOR LOCAL BLOOD FLOW MEASUR MENTS. Kurt Lewis Baum, USAF, 1982, 79 pages, Master of Science in Electrical Engineering, University of Illinois.

Tissue blood perfusion is a fundamental measurement in physiology that affects the entire spectrum of medical practice and research. A new and innovative method is under development by Dr. Kenneth R. Holmes and Dr. Michael M. Chen at the University of Illinois. Their thermal pulse-decay method utilizes a small thermistor to pulse heat the tissue under study. The thermistor is then used to record tissue temperature as the heat dissipates due to thermal conductivity and blood perfusion. From this cooling data, local blood perfusion can be calculated by various computer routines.

The process of initiating and controlling the experiment, aquiring and storing the data, and calculating perfusion parameters has been computer automated. The system is based on a Digital Equipment Corporation LSI 11 minicomputer. The software package developed for the system is user oriented. It can control up to six probes at once, performing both heating and measurement tasks. The user is free to choose the duration of the heat pulses, as well as the sampling rate and sampling duration after the heat pulse. The program automatically generates a data file for each active probe. The files can be recalled for display on a graphics terminal or for calculating perfusion parameters. In addition, an automatic mode is available which repetitiously performs the experiment with no user interaction.

The computer automation of the thermal pulse technique for local blood flow measurements will allow further development of this promising new measurement tool.

AFIT/CI/NR/82-45T

AFIT RESEARCH ASSESSMENT

The purpose of this questionnaire is to ascertain the value and/or contribution of research accomplished by students or faculty of the Air Force Institute of Technology (ATC). It would be greatly appreciated if you would complete the following questionnaire and return it to:

AFIT/NR
Wright-Patterson AFB OH 45433

RESEARCH TITLE: Computer Automation of the Thermal Pulse Technique for Local Blood Flow Measurements

AUTHOR: Kurt Lewis Baum

RESEARCH ASSESSMENT QUESTIONS:

1. Did this research contribute to a current Air Force project?

( ) a. YES ( ) b. NO

2. Do you believe this research topic is significant enough that it would have been researched (or contracted) by your organization or another agency if AFIT had not?

( ) a. YES ( ) b. NO

3. The benefits of AFIT research can often be expressed by the equivalent value that your agency achieved/received by virtue of AFIT performing the research. Can you estimate what this research would have cost if it had been accomplished under contract or if it had been done in-house in terms of manpower and/or dollars?

( ) a. MAN-YEARS _________ ( ) b. $_________

4. Often it is not possible to attach equivalent dollar values to research, although the results of the research may, in fact, be important. Whether or not you were able to establish an equivalent value for this research (3. above), what is your estimate of its significance?

( ) a. HIGHLY SIGNIFICANT ( ) b. SIGNIFICANT ( ) c. SLIGHTLY SIGNIFICANT ( ) d. OF NO SIGNIFICANCE

5. AFIT welcomes any further comments you may have on the above questions, or any additional details concerning the current application, future potential, or other value of this research. Please use the bottom part of this questionnaire for your statement(s).

NAME GRADE POSITION

ORGANIZATION LOCATION

STATEMENT(s):

82-...

COMPUTER AUTOMATION OF THE THERMAL PULSE

TECHNIQUE FOR LOCAL BLOOD FLOW MEASUREMENTS

BY

KURT LEWIS BAUM

B.S., United States Air Force Academy, 1981

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1982

Urbana, Illinois

COMPUTER AUTOMATION OF THE THERMAL PULSE TECHNIQUE FOR *LOCAL BLOOD FLOW MEASUREMENTS*. Kurt Lewis Baum, USAF, 1982, 79 pages, Master of Science in Electrical Engineering, University of Illinois.

Tissue blood perfusion is a fundamental measurement in physiology that affects the entire spectrum of medical practice and research. A new and innovative method is under development by Dr. Kenneth R. Holmes and Dr. Michael M. Chen at the University of Illinois. Their thermal pulse-decay method utilizes a small thermistor to pulse heat the tissue under study. The thermistor is then used to record tissue temperature as the heat dissipates due to thermal conductivity and blood perfusion. From this cooling data, local blood perfusion can be calculated by various computer routines.

The process of initiating and controlling the experiment, aquiring and storing the data, and calculating perfusion parameters has been computer automated. The system is based on a Digital Equipment Corporation LSI 11 minicomputer. The software package developed for the system is user oriented. It can control up to six probes at once, performing both heating and measurement tasks. The user is free to choose the duration of the heat pulses, as well as the sampling rate and sampling duration after the heat pulse. The program automatically generates a data file for each active probe. The files can be recalled for display on a graphics terminal or for calculating perfusion parameters. In addition, an automatic mode is available which repetitiously performs the experiment with no user interaction.

The computer automation of the thermal pulse technique for local blood flow measurements will allow further development of this promising new measurement tool.

ACKNOWLEDGEMENTS

I wish to express my sincere appreciation to my advisor, Dr. Richard L. Magin, for providing me the opportunity to work on this project. I am also grateful for his support and guidance throughout the past year.

A very special thanks is extended to Dr. Kenneth R. Holmes, without whom this project would not have been possible. His research goals and desires were the determining factors in the direction of this project.

TABLE OF CONTENTS

I. INTRODUCTION

Background

Tissue blood perfusion is a fundamental measurement in physiolgy that affects the entire spectrum of medical practice and research [1]. Many different techniques have been developed to measure tissue blood flow. Two common methods involve indicator dilution and radio-labeled microspheres [2]. These measurements are complicated and cannot be repeated at frequent intervals.

A different category of blood perfusion measurements involves the use of thermal techniques. These methods have the potential to overcome the limitations of the indicator dilution and microsphere techniques. However, most thermal methods have the drawback of comparing the heat dissipation against a value of thermal conductivity for nonperfused tissue. This requires either a suspension of the blood flow to the tissue or the use of tabulated values for the type of tissue being examined. Suspension of blood flow is traumatic to tissue and might mean sacrificing the animal, while using tabulated values can lead to inaccuracies due to assumptions made in creating the table. In addition, the theoretical basis for some of these methods is open to question, since the volume of tissue being sampled is not much larger than the probe itself, and probe size and shape have been shown to alter results [1].

The thermal pulse-decay method being developed at the University of Illinois by Dr. Kenneth R. Holmes and Dr. Michael M. Chen is a thermal method that overcomes all of the problems described above. This method consists of inserting a small thermistor into the tissue of interest. A known quantity of heat is deposited in the tissue when current flows through the thermistor. The thermistor is then used to measure post-pulse tissue cooling, from which local perfusion and thermal conductivity can be determined [1].

The basis of these calculations arises from the heat dissipation mechanisms working in the tissue. With no blood perfusion, the primary decay in temperature is due to thermal diffusion. This temperature decay can be mathematically modeled as a decaying power series [1]. Blood perfusion is a thermal transfer mechanism that can be modeled as an exponential decay of temperature. In vivo tissue cooling typically includes both of these mechanisms and can be modeled as a product of the exponential and power series. From the shape of this curve, the thermal conductivity and the local blood perfusion can be determined [1].

The advantages of this method are: it provides an absolute measurement of the volumetric perfusion rate (ml blood/ml tissue sec) without requiring calibrations or stop-flow measurements; the sampling volume is considerably larger than the volume of tissue traumatized by the microprobe; the probe shape and size do not affect the results; the electronics and

calculations are extremely simple; the increase in tissue temperature is usually only 0.5 C; the small diameter of the probe causes minimal trauma in the tissue under examination [3,4].

Some drawbacks of this method are that it does require insertion of a probe, it yields point information, and it is an indirect measurement process.

Problem and Scope

Small thermistor beads are fabricated into needle-like probes to aid in insertion into the tissue and minimization of trauma in the tissue. The thermistor is incorporated into a bridge circuit that performs both heating and measurement roles. The bridge output is used to generate cooling curves on a strip chart recorder. Computer programs have been written to analyse these cooling curves for thermal conductivity and perfusion rate.

The problem with this procedure is that the data must be read point by point from the the cooling curves and typed into the computer. This process is both time consuming and inaccurate. Additionally, since the data is analyzed after the experiment has been completed, the operator has no opportunity to modify the experimental parameters.

The solution to these problems is to automate the control of the experiment, the acquisition and storage of the data, and calculations performed on the data. This report describes a

computer based hardware and software system that will not only solve the problems of entering cooling curves into the computer and freeing the operator from continuous supervision of the system, but should also give the operator the advantage of having real-time blood flow data to tailor each succeeding measurement. Figure 1 is a block diagram of the proposed system.

Presentation

Chapter II describes the characteristics of the thermistor bridge circuit, the governing equations, and the desired specifications for the proper operation of the control system.

Chapter III describes the hardware used by the system while Chapter IV documents the system software.

Chapter V presents the results of experimental testing of the entire system.

Chapter VI contains the conclusions about the present system and recommendations for future systems.

Software flowcharts and a complete program listing are contained in the appendices.

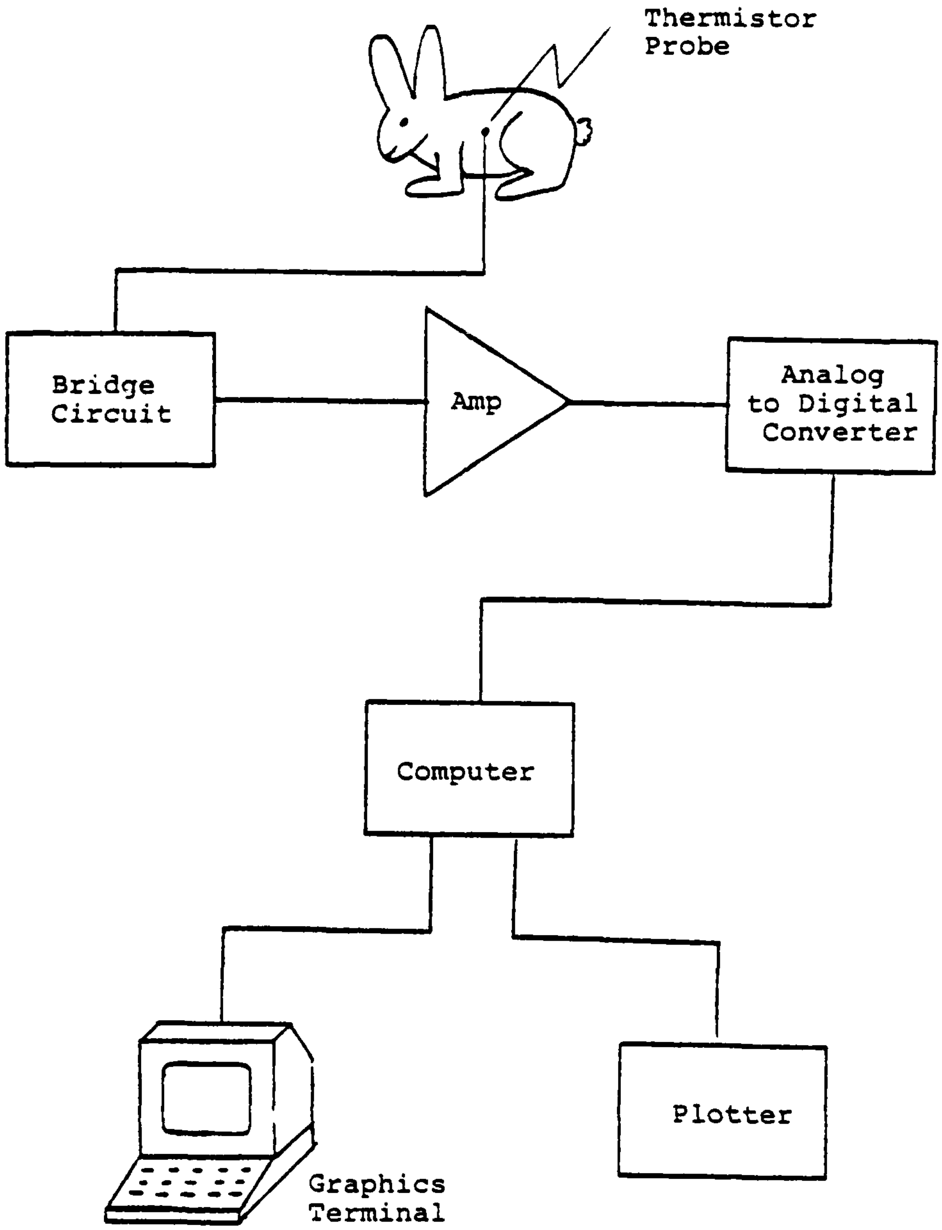


Figure 1. System Block Diagram

II. THERMAL PULSE-DECAY METHOD

Description

The bridge circuit containing the thermistor probe is a very simple balancing circuit with one modification that allows current to pass through the probe during heating. This circuit is shown in Figure 2. The balance resistor, R_b, is used to adjust the bridge output. A five Volt signal impressed at V_t can drive the transistor switch for the heater relay. The duration of this signal determines the length of the heating pulse and thus the energy delivered to the tissue.

Simple circuit analysis of the bridge yields the following relationship between the output voltage, E, and the probe resistance, R_p.

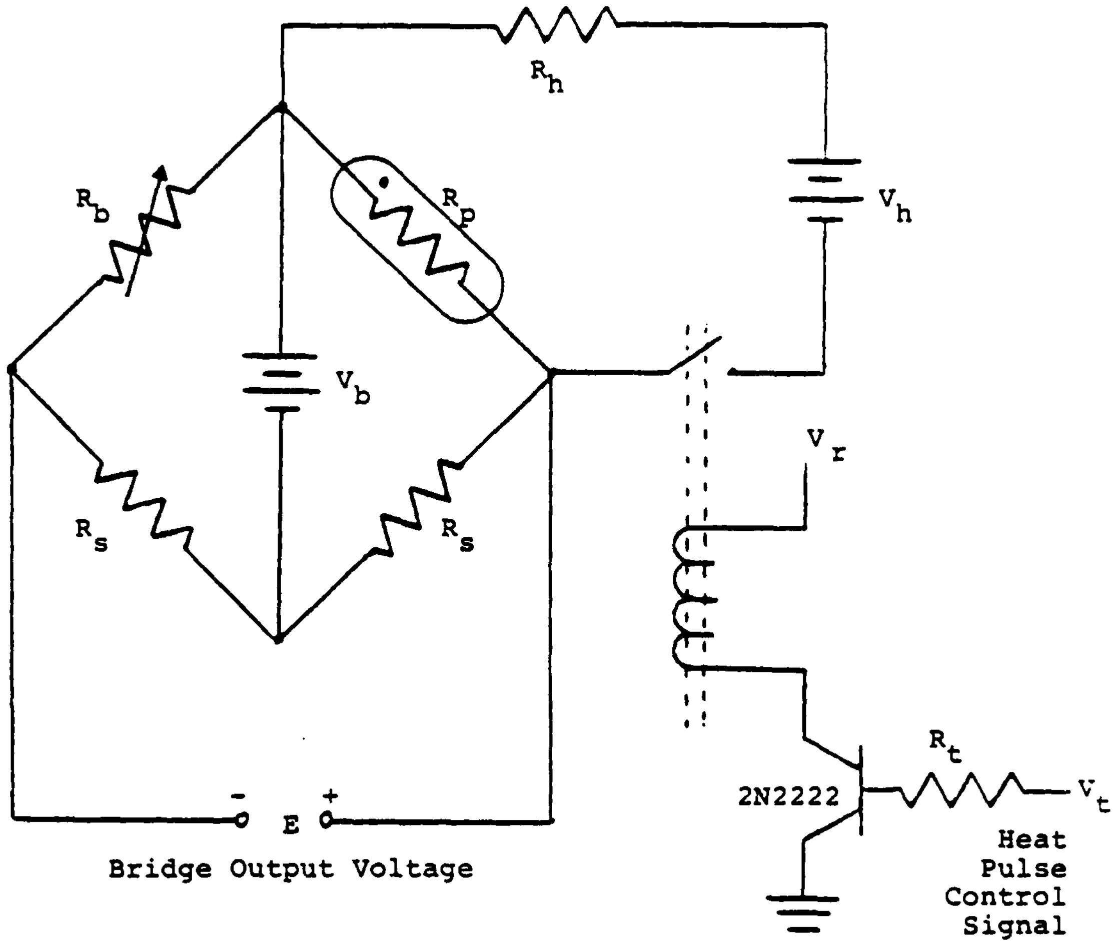
$$R_p = \left(\frac{R_s V}{R_s V/(R_b + R_s) + E} \right) - R_s \tag{1}$$

Then, once the resistance of the thermistor is known, the temperature, T, can be found as:

$$T = A - B \ln(R_p) \tag{2}$$

where A and B are calibrated constants of the particular thermistor probe in use [1].

The temperature rise due to tissue heating causes a corresponding increase in the output voltage. Thus, even though a nonlinear relationship exists between voltage and



V_h = 4.23 Volts

V_b = 5.45 Volts

V_r = 12 Volts

R_p = Thermistor

R_b = Bridge Balance Potentiometer (1000 to 1500 Ohms)

R_s = 22.05 kOhms

R_t = 3.3 kOhms

Figure 2. Thermistor Bridge Circuit

temperature, a cooling curve obtained by plotting the ouput voltage of the bridge looks similar to a cooling curve that plots actual temperature.

For a cooling curve expressed in Volts, the typical experiment will have about a seven millivolt range. In addition to the actual cooling part of the curve, important information is contained in the temperature values immediately prior to heating. This pre-sample period can be used to project a baseline for use in normalizing the cooling part of the curve to temperature drifts. A sample cooling curve is shown in Figure 3.

System Specifications

The following is a list of desired specifications for the automated control and data acquisition system.

1. Deliver a square wave five Volt pulse to each of six bridge circuits to control the heating cycle.
2. Vary the duration of each pulse length independently over a range of zero to twenty seconds in one-tenth second intervals.
3. The duration time of each heat pulse must be stored for computation purposes.
4. Monitor and record voltage output of six thermistor bridges with a resolution of three millivolts and a range of minus five to five volts.
5. Allow for sampling rates up to sixty Hertz in each of six channels.

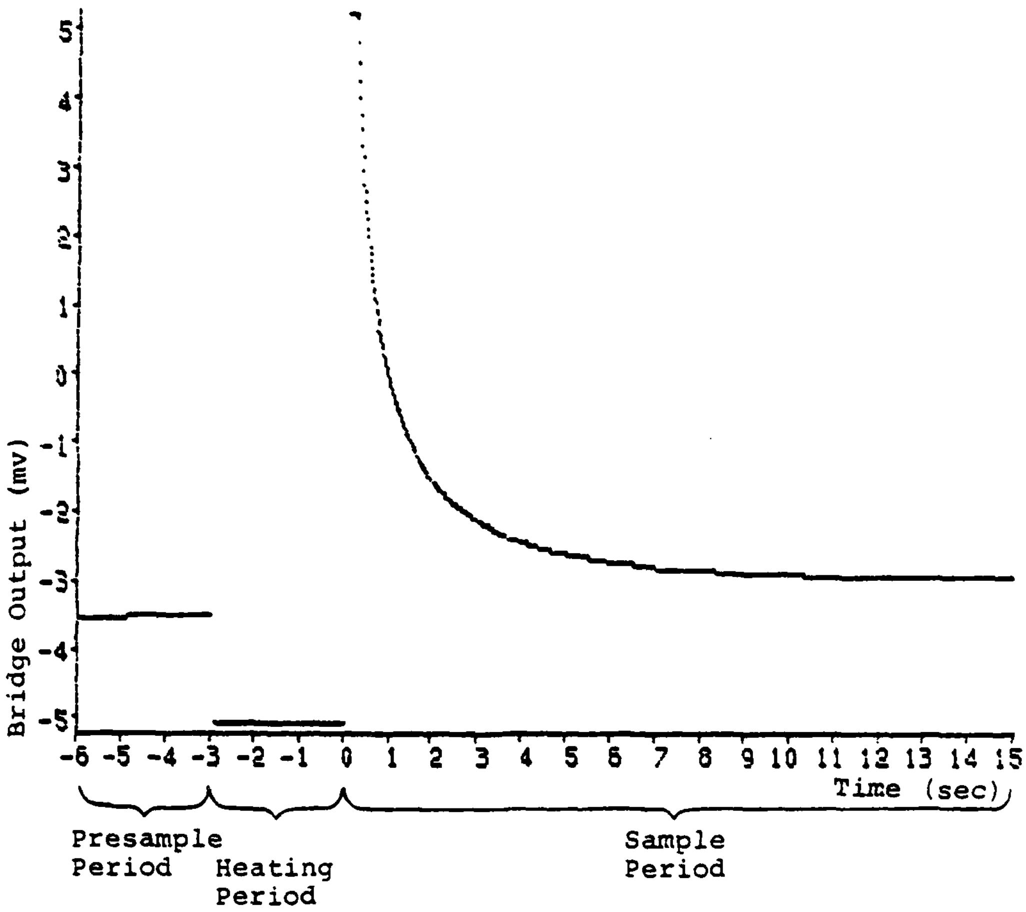


Figure 3. Typical Cooling Curve

6. Channel selection may be varied without loss of information. Channels need not be operated in sequential order.

7. Stored data may be recalled for display on a video terminal or printed on hard copy.

8. System must include a file management system for stored data files.

9. Calculations may be performed while system is not being used to measure perfusion.

10. Calculate and display steady state body temperature as calculated from thermistor data. Calibrate system for each thermistor.

11. If input voltage exceeds an operator selected value, system will issue an audible alarm. Alarm may be disabled.

12. Repeat measurements (delivery of pulses) automatically.

III. HARDWARE

The system hardware is composed of three main units. These are the thermistor bridge, the signal conditioner, and the computer. Figure 4 is a block diagram of the entire computer system, with the computer separated into its major components.

Bridge Circuit

The bridge circuit is described in Chapter II.

Signal Conditioning

Preliminary tests of the bridge circuit showed considerable noise to be present, which appeared in two basic forms. The most prevalent noise was sixty cycle power line interference, which had a magnitude up to four millivolts, almost as large as the desired signal. The second kind of noise was a very high frequency noise whose source could not be identified.

In addition to the noise, the signal strength was on the order of millivolts, too weak to be used as an input signal to the analog to digital converter, which measures in volts. The signal conditioning block, shown in Figure 5, is used to correct for these two problems. First, the signal was amplified by a differential laboratory amplifier with a gain of 1000. The input signal for a typical experiment has a range of about seven millivolts, with both positive and

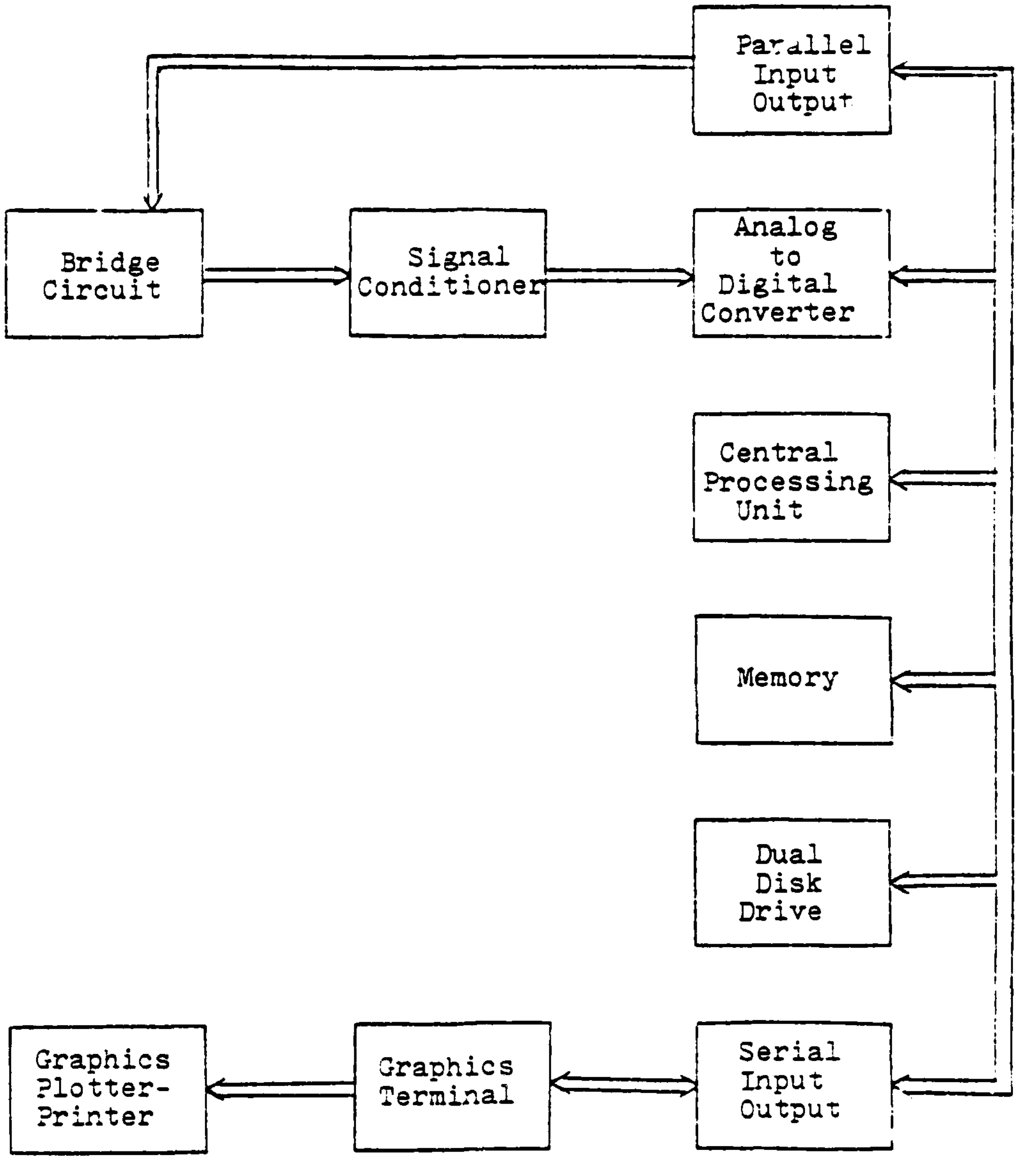


Figure 4. System Block Diagram

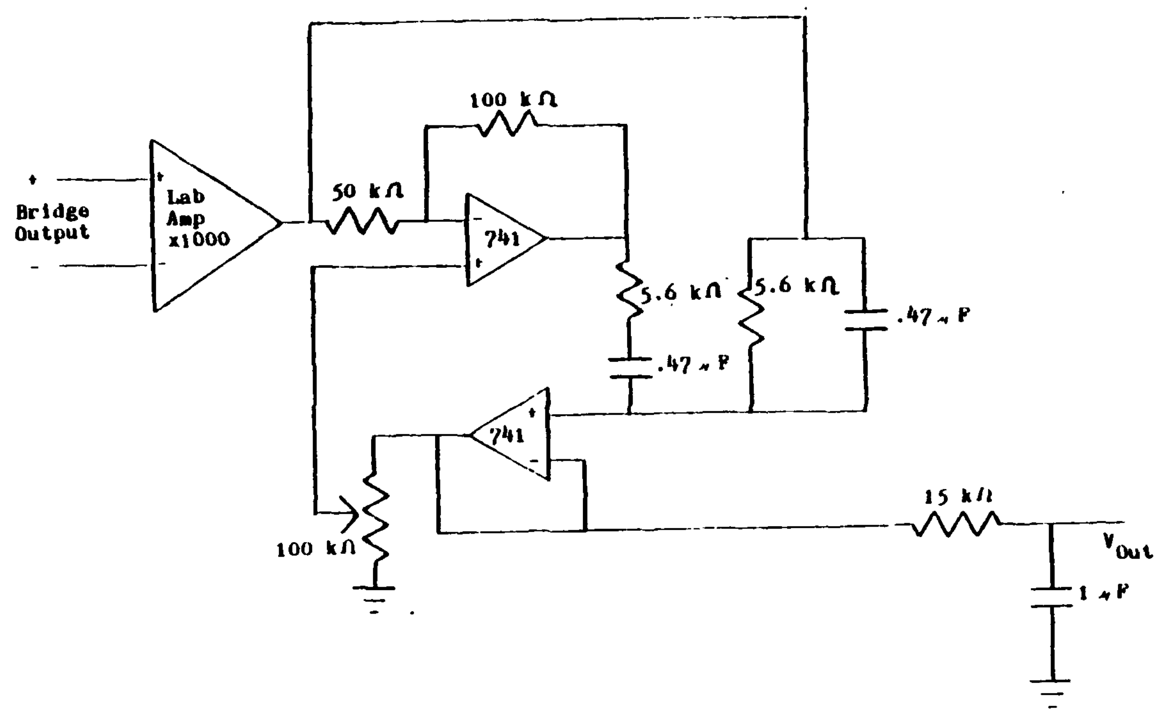


Figure 5. Signal Conditioning Schematic Diagram

negative components. The Analog to Digital Converter (ADC) has an input range of minus five to plus five volts. If the bridge is balanced at the beginning of an experiment for an output of about three millivolts, the gain of 1000 will provide a signal at the input of the ADC that will range from minus three to plus four volts. This is entirely within the range of the ADC and still allows for some temperature drift in between heating pulses.

A two stage filter was built to eliminate the noise problem. The first stage consists of a second-order band reject filter, the output of which provided 38 decibels of suppression centered at 60.4 Hertz with a quality factor of five. This filter effectively attenuates the sixty cycle interference. The second stage of the filter is a simple resistor-capacitor low pass filter with a 3 dB point of 10.6 Hertz. This filter eliminates the high frequency noise and further reduces the sixty cycle noise. The frequency content of the input signal is much less than 10 Hertz, so the filter does not attenuate the thermal pulse data signal.

Computer

This section will describe the capabilities and functions of each block of the computer. The computer is a Digital Equipment Corporation (DEC) LSI 11 minicomputer. Hardware specifications for the LSI 11 can be found in References [5] and [6].

The Parallel Input-Output unit is a DEC DRV11. It consists of 16 separate input and output lines, along with appropriate control lines. In this application the unit is used to control the heat pulses and light appropriate indicator lights on the bridge unit. In future versions, it can be used to monitor switch positions on the bridge unit. The pin assignments for this board are contained in Appendix A. More information on the operation of this board is contained in Reference [6].

The Analog to Digital converter is an ADAC 1030 which has eight differential inputs each with a range of minus five to plus five volts and a programmable gain of 1, 2, 5, or 10. In future versions of the bridge, this gain can be used as part of the 1000 gain of the signal conditioner. With the full gain of 1000, the ADC's 12 bit resolution is able to look at signals from minus five millivolts to five millivolts in 2.5 microvolt increments. This meets the original specification for resolution. Timing for the sampling of the ADC is accomplished with the real time clock of the LSI 11 computer. This allows sampling rates as high as sixty hertz for each of six channels. Additional information on the operation of the ADC can be found in Reference [7].

The Central Processing Unit is a DEC LSI 11/23 CPU with Memory Management Unit (MMU), model KDF11-AA, along with the floating point hardware option, model KEF11-AA. The MMU will allow full and efficient use of the Chrislin memory board.

The speed and power of the LSI 11/23 CPU with floating point should allow close to real time calculaton of the desired parameters after the data has been collected.

The Chrislin CI-1123 memory provides a full 256 kilobytes of random access memory. In addition to containing the operating program, it allows for storage of quite large data arrays during sampling that can be transferred later to disk.

The Data Systems Design 470 disk drive provides two megabytes of on line storage. Drive one will be used as a system disk containing the source program and monitor, while drive two will be used to interchange data disks.

The serial interface is a DEC DLV11-J, which has four independent communication channels. One of these is used for the system terminal. The others could be used for a modem, a remote terminal, or a line printer.

The system terminal is a Lear Siglar ADM5 with a 512 retrographics board. This provides graphics capability that is software compatible to the Techtronics 4010 terminal. The main use of the graphics will be to display various cooling curves for visual inspection. Additional information is contained in References [8] and [9].

The GP100 graphics plotter is connected directly to the ADM5 terminal. It provides hardcopy for any of the graphics sent to the terminal or functions as a line printer for listing programs or data sets. It is further described in Reference [10].

IV. SOFTWARE

The system software is written in FORTRAN and resides on the system disk. Both the user and the operating program have access to an extensive library of programs written for the DEC LSI 11 computer system. These include the RT11 monitor, a text editor, a disk handler, linkers, compilers, various input-output routines, and a system library of FORTRAN callable subroutines. A full description of these programs is contained in Reference [11].

The FORTRAN operating program contained on the system disk allows the user to interact with the system to configure the experiment and data collection in an easy and flexible manner. In addition, it allows calculation routines to be performed on the data and several modes of automatic operation. The user also has the ability to list or graph any data file.

This chapter contains a complete description of the operating program. Figure 6 is a simplified flowchart of the main program. Flowcharts of the subroutines and a complete listing of the FORTRAN code are contained in Appendix C.

Main Program

The program begins with an initialization routine that sets each variable to a valid starting value. As the program is used, the initial values can be modified to reflect more accurately the desired initial conditions for the experiments.

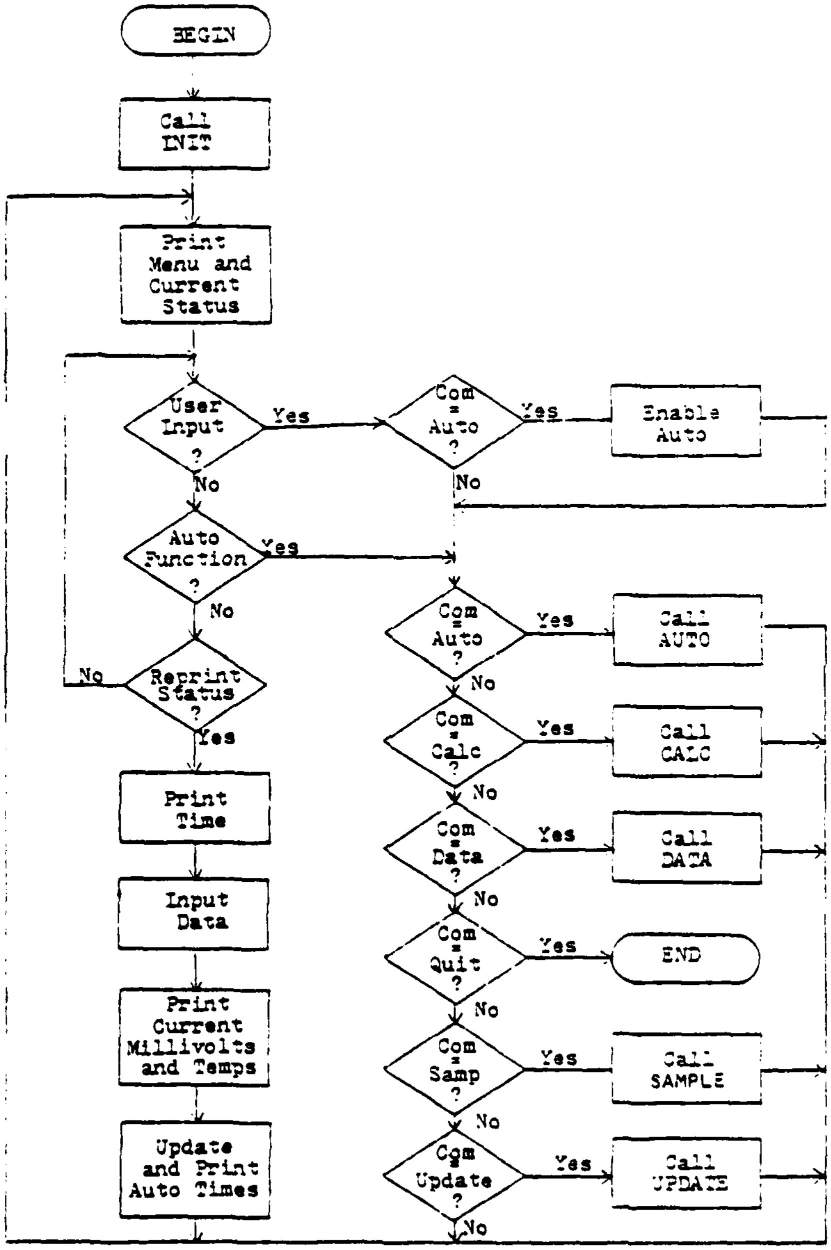


Figure 6. Simplified Flowchart of the Main Operating Program

The terminal is then filled with the current values of the program variables and a list of the commands available to the user. An example of this is shown in Figure 7. The commands are single letters that correspond to the function or parameter to its right. Each second, the current time is displayed, all six bridge outputs are sampled and displayed in both millivolts and degrees Celsius, and the time until the next automatic operation is updated and displayed. While in this mode, the software continuously checks for user input or the conditions necessary to cause an automatic operation.

Each user input causes the main program to call an appropriate subroutine which actually performs the command. A summary of each of the available commands is shown in Table I. The following is a description of each command along with its use and restrictions.

Command ´O´

This command allows the user to choose which channels will be active when either SAMPLE or CALCULATE is performed. Each channel can be independently turned on or off with no restrictions as to which channels must be used for a given number of probes. The software for command ´O´ is contained in subroutine UPDATE.

Command ´F´

This command allows the filename of any channel to be changed. Each filename must be constructed of three

```
TIME  13: 3: 16  DATE  5/16/82   DATA VERSION ONE

   CHANNEL                       1         2         3         4         5         6
D  SAMPLE STATUS                ON        ON
   CALCULATE STATUS             ON
F  FILENAME                 AAA000    AAB000    AAC000    AAD000    AAE000    AAF000
P  PROBE                        24        25         0         0         0         0
   CURRENT MILLIVOLTS        0.003     0.100     0.944     1.529     2.288     5.250
   CURRENT TEMPERATURE      34.432    35.585     0.000     0.000     0.000     0.000
   LAST TEMP                34.432    35.822     0.000     0.000     0.000     0.000
H  HEAT   (SEC:TICKS)         2: 0      2: 0      2: 0      2: 0      2: 0      2: 0
                  DURATION (SEC:TIC)  FREQUENCY (HZ)   READINGS   PERIOD
R  PRESAMPLE              3: 0                60           180          1
   SAMPLE                15: 0                60           900          1

L  UPPER ALARM LIMIT (MV)           6.23     A  AUTOMATIC RUN
   LOWER ALARM LIMIT (MV)          -6.00     B  BREAK FROM AUTO
E  EXPERIMENT INTERVAL (MIN:SEC)   0:45      C  CALCULATE
   EXPERIMENT REPETITIONS              5     D  LOOK AT DATA FILE
   NEXT EXPERIMENT  (MIN:SEC)      0: 0      S  SAMPLE
                                             Z  QUIT

PLEASE ENTER YOUR COMMAND
```

Figure 7. Video Terminal Presentation of Program Status and Commands

Table 1. Summary of Commands

Command	Description
O	Change Channel Activity Status
F	Change Data Filename
P	Change Probe, Bridge Balance, or Descriptive Text
H	Change Heat Pulse Duration
R	Change Duration and Frequency of Sampling
L	Change Alarm Limits
E	Change Experiment Repetition Parameters
A	Enter Automatic Operation
B	Break From Automatic Operation
C	Perform Calculations
D	Display Data Files
S	Perform Heat Pulse and Sampling
Z	Exit Program

alphanumeric characters followed by three numeric characters. This allows the user to describe the experiment using three letters and then determine the experiment repetition using the three numerals remaining. This filename is used by both SAMPLE and CALCULATE. Subroutine UPDATE performs the command 'F'.

Command 'P'

This command allows the user to change the probe number, probe calibration data, bridge balance conditions, and descriptive text of any channel. When a probe is changed, the software reads in the calibration data for the new probe from a disk file named PROBE.DAT. This file is maintained by the auxiliary program PROBEC.FOR which is described later. The calibration data is part of the data file stored when a sample is performed. It is needed to calculate absolute temperature.

Changing the bridge balance conditions requires the user to enter the new resistance of the potentiometer. This value is also needed for absolute temperature calculation, and is part of the data file stored by SAMPLE.

The descriptive text can be used to record probe placement, experiment objective or other pertinent information. It is also part of the data file stored by SAMPLE.

Subroutine UPDATE performs the command 'P'.

Command 'H'

This command allows the user to modify the duration of the heat pulse that is applied to the probe. Each channel can be varied independently in increments of one-sixtieth of a second. A duration of zero is allowed and simply means that that channel will not be pulsed during sampling. This allows probes to be used to determine heat patterns delivered by other probes. Subroutine UPDATE performs the command 'H'. The heat pulse duration is part of the data file stored by SAMPLE.

Command 'R'

This command allows the user to define the sampling rate and duration for an experiment. The samples taken before the heat pulse, the presample period, can be configured independently of the samples taken after the heat pulse. The user inputs the desired time duration and frequency of sampling. The program calculates and displays the actual number of readings to be taken and the time period (in one-sixtieth of seconds) between samples. These values are part of the data file stored by SAMPLE. Subroutine UPDATE performs the command 'R'.

Command 'L'

Thiscommand lets the user set limits for an audible alarm that monitors the input voltages of the channels that are active for sampling. If the input voltage is not within the specified range, the program sounds the bell on the terminal

each second. With the correct limits, the alarm can be used to notify the user when a bridge needs rebalancing before starting an experiment. Subroutine UPDATE performs the command ´L´.

Command ´E´

This command allows the user to define the parameters for operating under automatic control. The time interval between experiments, the number of experiments to be performed and the time until the first experiment can all be set to any value. Subroutine UPDATE performs the command ´E´.

Command ´A´

This command enables the automatic operation of the program, as defined by the automatic control parameters. It is performed by the main program and subroutine AUTO. When called, subroutine AUTO will call SAMPLE and CALCULATE as needed, increment the filenames of the active channels and update the automatic operation parameters. Using the ´A´ command, the operator can free himself from having to continuously monitor and initiate experiments or calculations.

Command ´B´

This command disables automatic operation of the program. The time until the next experiment will continue to count down, but AUTO will not be called if the count reaches zero.

Using the 'B' command, the user can exit automatic control, change a system parameter, reenter automatic control and not lose the correct spacing between experiments. The main program performs the 'B' command.

Command 'C'

This command performs the desired calculation on each channel that is active for calculate. Subroutine CALC performs this command but does not perform the calculations. CALC reads and prepares the data file for a subroutine called CRUNCH, which is to perform the actual calculations. CRUNCH can either be a thermal conductivity or blood perfusion routine that can be linked to CALC for use.

Command 'D'

This command allows the user to examine a data file by listing or graphing. Listing a data file consists of labeling, formatting and printing the system configuration at the time of sampling and the entire presample and sample data.

The graphics package plots the cooling curve and labels it with pertinent information from the data file. The user has the options of changing the size and position of the graph as well as expanding the time scale to display only a portion of the cooling curve.

Subroutine DATA performs the command 'D'.

Command ´S´

This command causes the computer to perform the sampling as defined by the system parameters. For each active channel the appropriate presample is collected, the probe is heated, the desired sample data is collected, and a data file is created on disk. The format of the data file is in appendix B. It consists of all possible information that might be desired at a later time to analyze the results of the experiment. Subroutine SAMP performs the command ´S´.

Command ´Z´

This command terminates the operation of the program and returns control to the RT11 monitor.

Auxiliary Program

The auxiliary program PROBEC.FOR is used to maintain the data file that contains the current probe calibration constants. The user has the option of entering calibration data for a new probe or changing calibration data for an existing probe in the file. Each time new data is entered into this file, the user can also enter the date of calibration. The probe calibration and the date are stored in this for use by the system software. Specifications for the data file PROBE.DAT are contained in appendix B.

V. RESULTS

The data aquisition system has been successfully used in a variety of configurations.

Figures 8 through 14 are actual output from the system. Figure 8 shows the cooling curve of a probe that was placed in a 100 ml solution of glycerin at 38 degrees Celsius. The presample data is shown from minus five to minus two seconds. The plot disappears from minus two to zero seconds, since that is during the heat pulse and the bridge output is not valid. The sample data is shown from zero to fifteen seconds.

This curve shows that the probe, the bridge, the signal conditioning, and the analog to digital converter all worked properly to deliver the appropriate data to the computer. It also shows that the software is capable of procuring the data, storing it, and retrieving it for display.

Figure 9 is a cooling curve from a similar experiment but the heating pulse has been reduced to only one-sixth of a second. The curve decays much more rapidly since less heat was deposited by the probe. In order to display this data in a more visible manner, the software allows for expansion of the time scale so that any portion of the curve can be shown. An example of this capability is shown in Figure 10 using the same data file as Figure 9.

Figure 11 is from a data file that was created from an experiment using a live rabbit. The blood perfusion in the

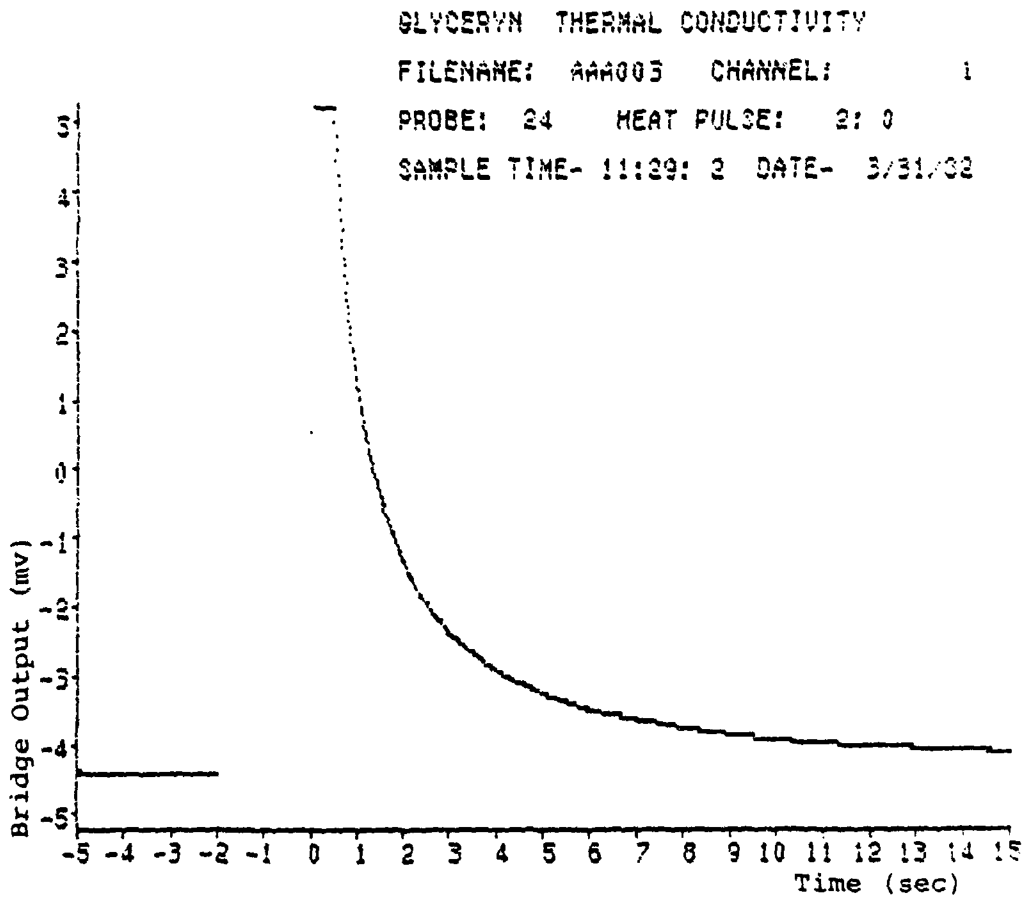


Figure 8. Thermal Conductivity Cooling Curve

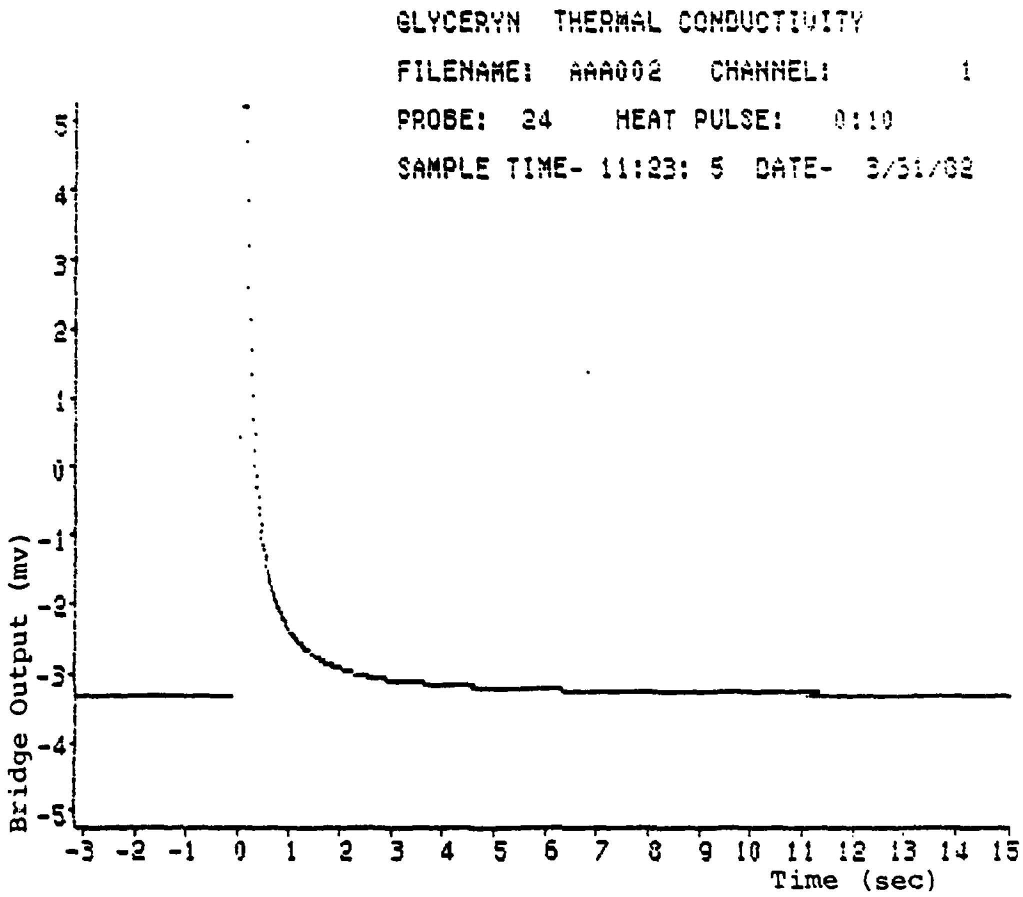


Figure 9. Thermal Conductivity Cooling Curve

Short Heat Pulse Duration

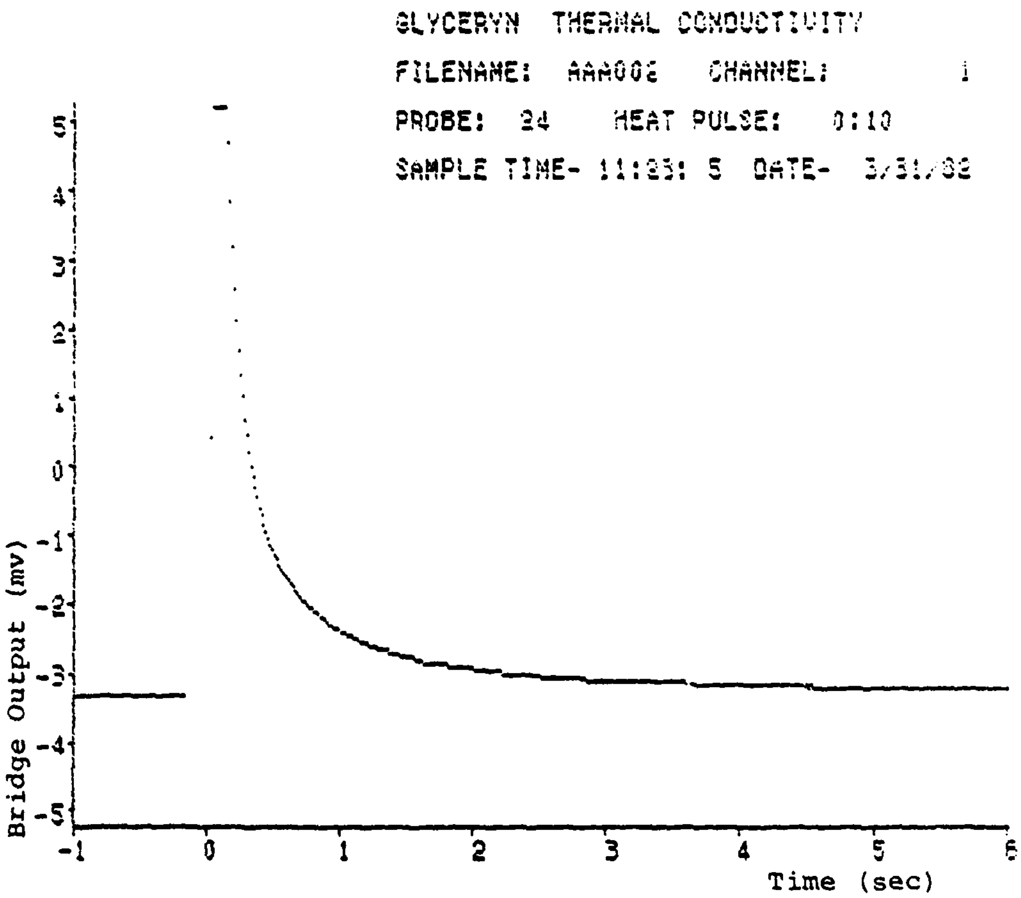


Figure 10. Thermal Conductivity Cooling Curve
Expanded Time Scale

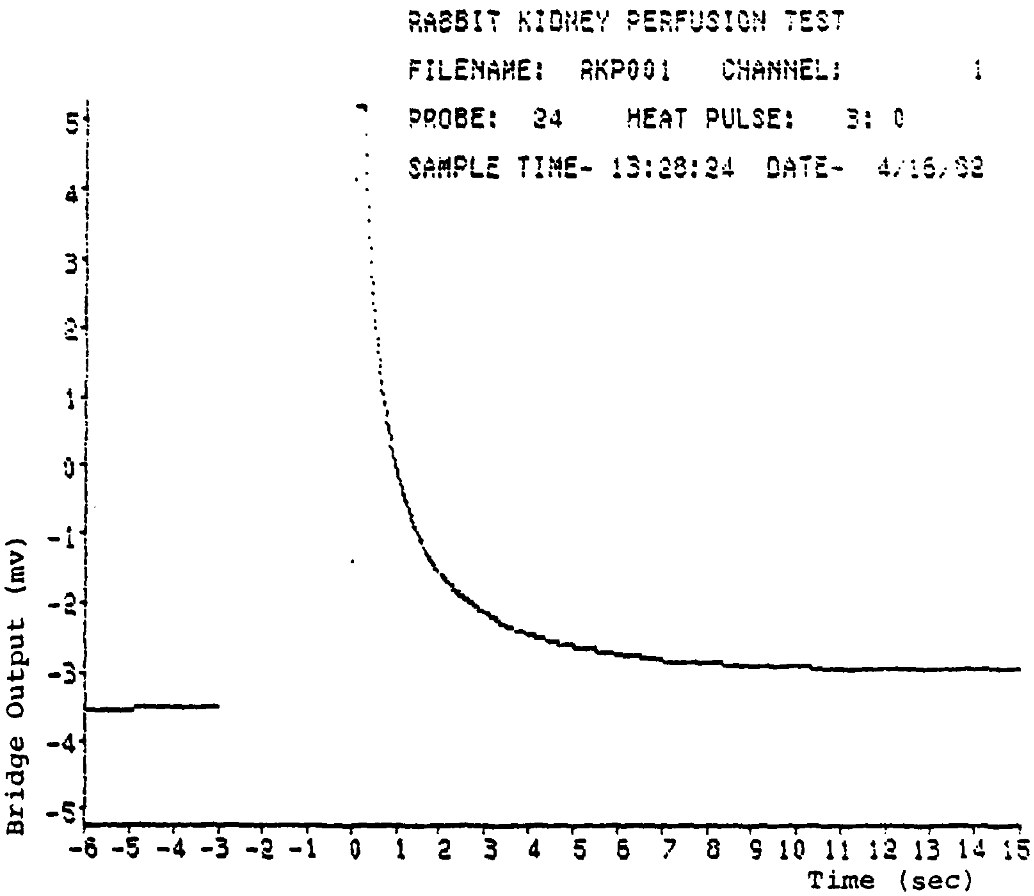


Figure 11. Kidney Blood Perfusion Cooling Curve

kidney could be determined from this data.

Figures 12 and 13 are from data files created from experiments conducted on the liver of a live rabbit. The probe for Figure 12 was placed very close to a large blood vessel in the liver. This results in a cooling curve that decays quite rapidly. Also present on this curve is some respiratory artifact that causes the curve to rise and fall periodically. Figure 13 is from a test on the same liver but the probe has been moved away from any blood vessels. Perfusion is still present, but at a much lower rate and the respiratory artifact is no longer visible. In order to compare these two curves, Figure 14 was generated by expanding and overlaying the data from the two files on the same plot. The difference between the high perfusion curve and the low perfusion curve is now seen as different rates of decay in the cooling curves. The higher perfusion near the large blood vessel rapidly removes heat from the vicinity of the probe, resulting in a faster decay than is seen in the low perfusion case.

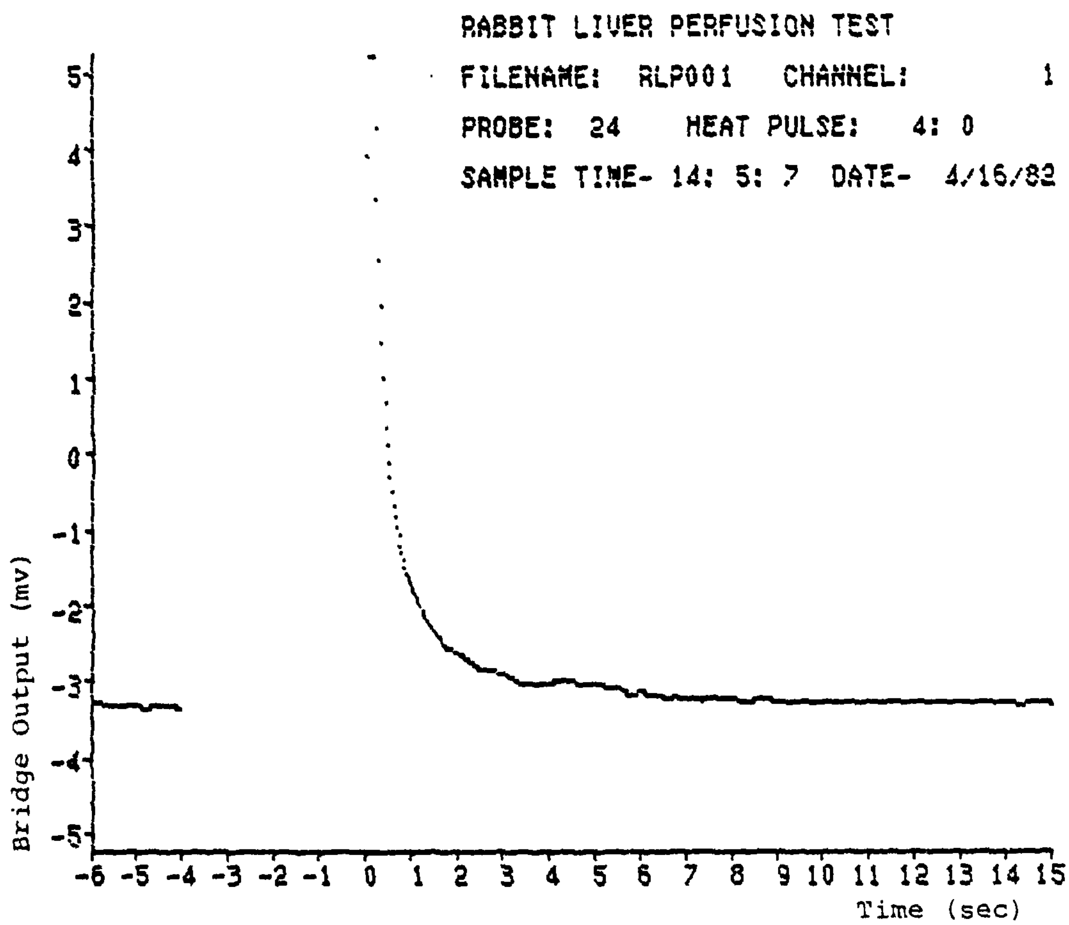


Figure 12. Rabbit Liver, High Blood Perfusion

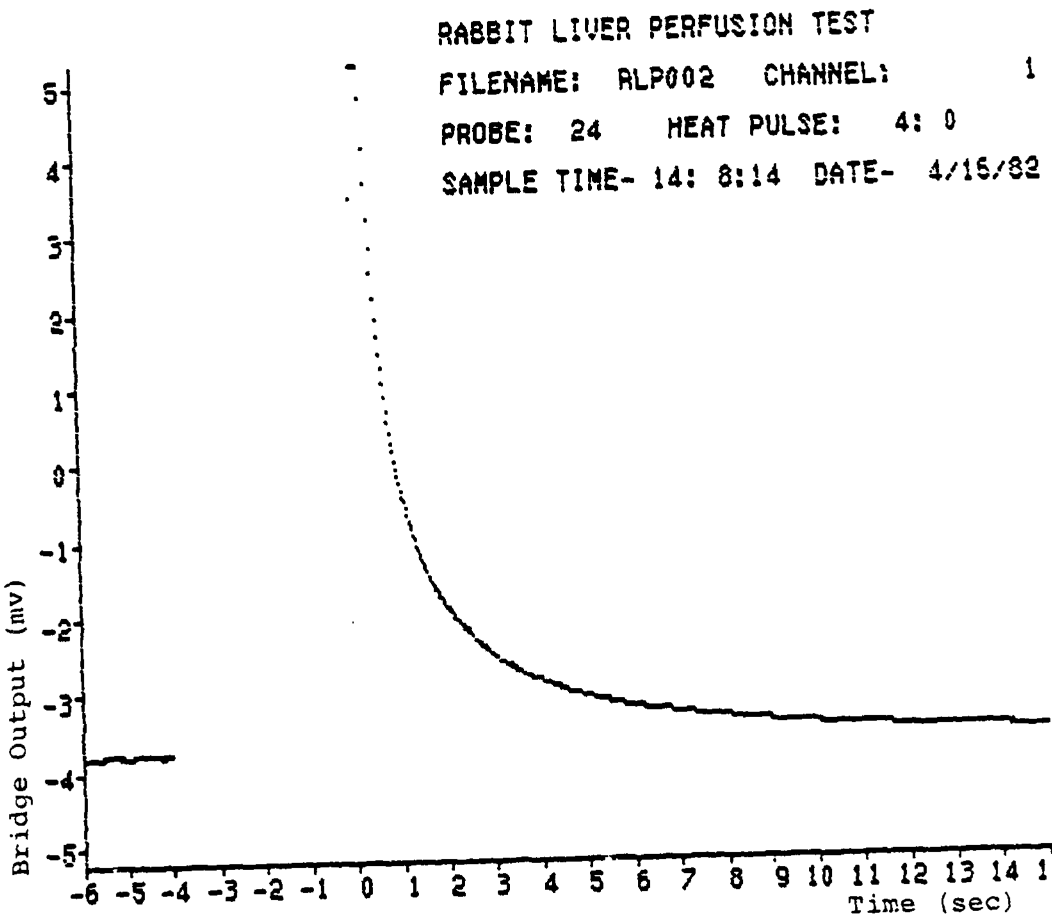


Figure 13. Rabbit Liver, Low Blood Perfusion

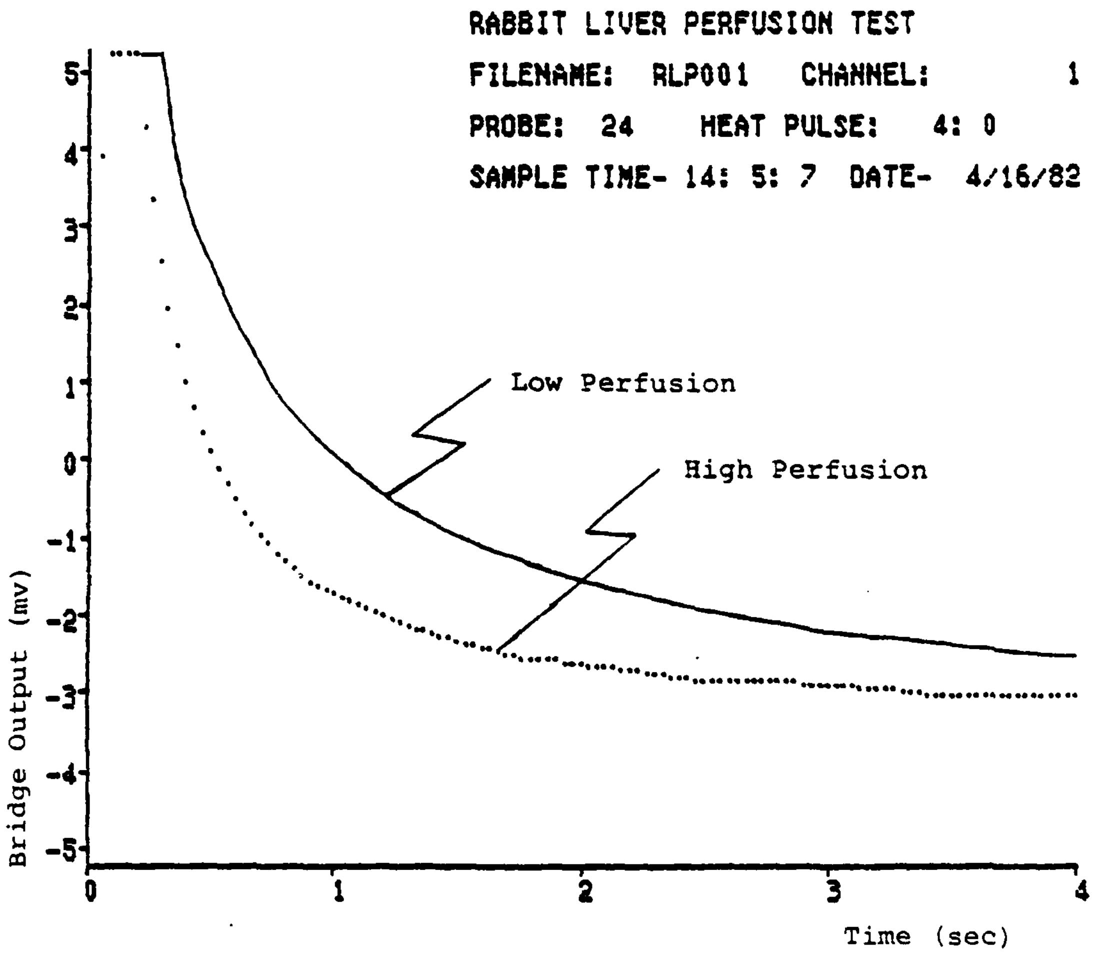


Figure 14. Rabbit Liver, Comparison of High and Low Perfusion

VI. CONCLUSIONS AND RECOMMENDATIONS

Conclusions

An automated control and data aquisition system for the thermal pulse-decay method of blood perfusion measurement has been designed and tested. The system meets or exceeds the desired specifications, providing a useable tool for the continuation of research in local blood perfusion measurement.

Recommendations

Based on system performance until this point, the following recommendations are proposed as future modifications to the system. Additional desired modifications will become apparent as the system is utilized more frequently.

1) Incorporate into the auxiliary program, PROBEC.FOR, the capability to automatically calibrate the thermistor probes.

2) Enhance the data presentation, particularly the ability to show more than one cooling curve on the same plot.

3) Provide more disk file security to prevent accidental overwriting of data.

4) Provide for easy transfer of data files to or from this computer and another.

5) Incorporate computer routines to analyse the cooling curves for both thermal conductivity and blood perfusion.

APPENDIX A
SYSTEM HARDWARE CONNECTIONS

This appendix contains the wiring list for connecting the DEC DRV11 parallel input-output board to the bridge circuit. Two twenty-five wire ribbon cables carry the signals listed below in Table 2.

Table 2. Pin Assignments for Bridge Control

Signal	Channel	Connector, Pin
Presample LED	1	J1, 11
Presample LED	2	J1, 12
Presample LED	3	J1, 13
Presample LED	4	J1, 14
Presample LED	5	J1, 15
Presample LED	6	J1, 16
Heat Pulse and LED	1	J2, 11
Heat Pulse and LED	2	J2, 12
Heat Pulse and LED	3	J2, 13
Heat Pulse and LED	4	J2, 14
Heat Pulse and LED	5	J2, 15
Heat Pulse and LED	6	J2, 16
Sample LED	ALL	J1, 17

APPENDIX B

DATA FILE STRUCTURES

This appendix contains the structure of the two types of data files used by the operating software. PROBE.DAT contains the calibration data for the thermistor probes and is outlined in Table 3. The structure of the data files created by SAMPLE is outlined in Table 4.

Table 3. PROBE.DAT Structure

Record	Variable Description	Disk Format
First	Number of Probes	I*2
Second	Probe Number	I*2
	Month	I*2
	Day	I*2
	Year	I*2
	A	R*4
	B	R*4
	RB0	R*4
	BETA	R*4

Repeating for each probe in the data file.

Table 4. Data File Structure

Variable Description	Variable Name	Disk Format
Data Version	1	I*2
Time	ISTIM	I*4
Date	KDATE	3 I*2
Channel	I	I*2
Text	ITEXT	20 I*2
Active Files	IFILE(1)	3 I*2
Active Files	IFILE(2)	3 I*2
Active Files	IFILE(3)	3 I*2
Active Files	IFILE(4)	3 I*2
Active Files	IFILE(5)	3 I*2
Active Files	IFILE(6)	3 I*2
Probe Number	IPROBE	I*2
RBO	PPRBO	R*4
BETA	PPBETA	R*4
A	PPA	R*4
B	PPB	R*4
Probe Calibration Date	ICDATE	3 I*2
Bridge Balance Resistance	IPPRBB	I*2
Bridge Balance Time	IPPTBB	I*4
Bridge Balance Voltage	PPVBB	R*4
Heat Pulse Duration (Ticks)	IHEAT	I*2
Presample Readings	IPSR	I*2
Presample Period (Ticks)	IPSP	I*2
Sample Readings	ISR	I*2
Sample Period (Ticks)	ISP	I*2
Presample Data	IDATA	(IPSR) I*2
Sample Data	IDATA	(ISR) I*2

APPENDIX C

OPERATING PROGRAM

Figure 6 in Chapter IV is a flowchart of the main program. This appendix contains a listing of the FORTRAN code of the main program followed by flowcharts and code listings of the subroutines.

Figure 15 shows the flowchart of subroutine INIT (p. 46),
Figure 16 shows the flowchart of subroutine UPDATE (p.49-50),
Figure 17 shows the flowchart of subroutine SAMP (p. 57),
Figure 18 shows the flowchart of subroutine CALC (p. 63),
Figure 19 shows the flowchart of subroutine AUTO (p. 65),
Figure 20 shows the flowchart of subroutine DATA (p. 68),
Figure 21 shows the flowchart of PROBEC.FOR (p. 74), and
Figure 22 shows the flowchart of subroutine GET (77).

```
C          THIS IS THE MAIN PROGRAM FOR THERMAL PULSE
C          STUDIES.
C          IT WAS WRITTEN BY  KURT L. BAUM

C   * * *  COMMON VARIABLES  * * *
           LOGICAL*1          HAZ(3),HAZA(5)
           INTEGER            I2(256)
           INTEGER*4          I4(12)
           REAL*4             R4(64)
           INTEGER            IDATA(1800)

           COMMON   /HAZEL/HAZ,HAZA
           COMMON   /VARBLE/I2,I4,R4
           COMMON   /TEMPS/IDATA
C   * * *  END COMMON BLOCK  * * *

C   * * *  BEGIN COMMON DESIGNATIONS  * * *
           INTEGER IACTVS(6),IACTVC(6),IPROBE(6),IHEAT(6),IPPRBB(6)
           INTEGER IFILE(6,3),ITEXT(6,20),ICDATE(6,3)
           INTEGER*4 IPPTBB(6)
           REAL*4 PPRB0(6),PPBETA(6),PPA(6),PPB(6),PPVBB(6)
           REAL*4 CTEMP(6),RLTEMP(6)

           EQUIVALENCE      (ICOM,I2(1)),            (IACTVS(1),I2(2))
           EQUIVALENCE      (IPROBE(1),I2(8)),       (IHEAT(1),I2(14))
           EQUIVALENCE      (IPPRBB(1),I2(20)),      (IPSR,I2(26))
           EQUIVALENCE      (IPSP,I2(27)),           (ISR,I2(28))
           EQUIVALENCE      (ISP,I2(29)),            (ITEXT(1,1),I2(30))
           EQUIVALENCE      (ICDATE(1,1),I2(150)),   (IACTVC(1),I2(168))
           EQUIVALENCE      (IFILE(1,1),I2(174)),    (IAUTOC,I2(192))
           EQUIVALENCE      (IEXREP,I2(193))
           EQUIVALENCE      (IPPTBB(1),I4(1))
           EQUIVALENCE      (PPRB0(1),R4(1)),        (PPBETA(1),R4(7))
           EQUIVALENCE      (PPA(1),R4(13)),         (PPB(1),R4(19))
           EQUIVALENCE      (PPVBB(1),R4(25)),       (CTEMP(1),R4(31))
           EQUIVALENCE      (RLTEMP(1),R4(37)),      (RUALIM,R4(43))
           EQUIVALENCE      (RLALIM,R4(44)),         (AUTOTN,R4(45))
           EQUIVALENCE      (AUTOTU,R4(46)),         (EXINT,R4(47))
C   * * *  END COMON DESIGNATIONS  * * *

C   * * *  BEGIN LOCAL VARIABLES  * * *
           LOGICAL*1 PRNTIM(9)
           INTEGER JTEMP(6)
           INTEGER ITYPE(30)
           INTEGER*4 KTIME
           INTEGER KTIM(2)
           REAL*4 CVOLT(6),TEMP

           EQUIVALENCE (KTIME,KTIM(1))
C   * * *  END LOCAL VARIABLES  * * *

C   * * *  BEGIN CODE SEGMENT  * * *
           HAZ(2)="200
           HAZ(3)="200
           HAZA(1)=27
           HAZA(2)=61
           HAZA(5)="200
           CALL INIT
           PRNTIM(9)="200

C   * * *  PRINT PROGRAM IDENTIFICATION  * * *
           TYPE 30
           HAZ(1)=25
           CALL PRINT(HAZ)
           HAZA(3)=40
```

```
      HAZA(4)=45
      CALL PRINT(HAZA)
      TYPE 1
1     FORMAT('THERMAL PULSE DECAY CONTROLLER')
      TYPE 30
      TYPE 2
2     FORMAT(T15,'WRITTEN BY KURT L. BAUM')
      ACCEPT 3
3     FORMAT(A)

C  * * *  PRINT SMORGASBOARD  * * *
5     TYPE 30
      HAZ(1)=26
      CALL PRINT(HAZ)
      CALL IDATE(KMON,KDAY,KYEAR)
      ENCODE(8,100,PRNTIM) KMON,KDAY,KYEAR
100   FORMAT(I2,'/',I2,'/',I2)
      CALL PRINT('TIME')
      HAZA(3)=32
      HAZA(4)=52
      CALL PRINT(HAZA)
      CALL PRINT('DATE')
      HAZA(4)=57
      CALL PRINT(HAZA)
      CALL PRINT(PRNTIM)
      HAZA(4)=77
      CALL PRINT(HAZA)
      CALL PRINT('DATA VERSION ONE')
      TYPE 105,1,2,3,4,5,6
105   FORMAT('    CHANNEL                 ',6(7X,I1))
      DO 107 I=1,6
      JTEMP(I)=IACTVS(I)
      IF(IACTVS(I).NE.0) GOTO 107
      JTEMP(I)='  '
107   CONTINUE
      TYPE 110,JTEMP
110   FORMAT('O   SAMPLE STATUS          ',6(6X,A2))
      DO 112 I=1,6
      JTEMP(I)=IACTVC(I)
      IF(IACTVC(I).NE.0) GOTO 112
      JTEMP(I)='  '
112   CONTINUE
      TYPE 115,JTEMP
115   FORMAT('    CALCULATE STATUS       ',6(6X,A2))
      TYPE 120,((IFILE(I,K),K=1,3),I=1,6)
120   FORMAT('F   FILENAME               ',6(2X,3A2))
      TYPE 135,IPROBE
135   FORMAT('P   PROBE                  ',6I8)
      TYPE 136
136   FORMAT('    CURRENT MILLIVOLTS')
      TYPE 137
137   FORMAT('    CURRENT TEMPERATURE')
      TYPE 138,RLTEMP
138   FORMAT('    LAST TEMP              ',6F8.3)
      TYPE 140,(IHEAT(I)/60,IHEAT(I)-IHEAT(I)/60*60,I=1,6)
140   FORMAT('H   HEAT    (SEC:TICKS)    ',6(I5,':',I2))
      TYPE 142
142   FORMAT('                    DURATION (SEC:TIC)   FREQUENCY (HZ)',
     1 '      READINGS    PERIOD')
      KTIM(1)=0
      KTIM(2)=IPSR*IPSP
      CALL CVTTIM(KTIME,KHRS,KMIN,KSEC,KTICKS)
      TYPE 145,KSEC,KTICKS,60/IPSP,IPSR,IPSP
145   FORMAT('R   PRESAMPLE          ',I2,':',I2,'
     1        I2,'              ',I4,'          ',I2)
```

```
      KTIM(2)=ISR*ISP
      CALL CVTTIM(KTIME,KHRS,KMIN,KSEC,KTICKS)
      TYPE 150,KSEC,KTICKS,60/ISP,ISR,ISP
150   FORMAT('    SAMPLE              ',I2,':',I2,'
     1        I2,'              ',I4,'           ',I2)
      TYPE 30
      TYPE 151,RUALIM
151   FORMAT('L   UPPER ALARM LIMIT (MV)          ',F5.2,
     1        T45,'A  AUTOMATIC RUN')
      TYPE 152,RLALIM
152   FORMAT('    LOWER ALARM LIMIT (MV)          ',F5.2,
     1        T45,'B  BREAK FROM AUTO')
      KTIM(1)=INT(EXINT/65536.0)
      KTIM(2)=INT(EXINT-65536.0*KTIM(1))
      CALL CVTTIM(KTIME,KHRS,KMIN,KSEC,KTICKS)
      TYPE 155,KMIN,KSEC
155   FORMAT('E   EXPERIMENT INTERVAL (MIN:SEC)   ',I2,':',I2,
     1        T45,'C  CALCULATE')
      TYPE 160,IEXREP
160   FORMAT('    EXPERIMENT REPETITIONS            ',I4,
     1        T45,'D  LOOK AT DATA FILE')
      TYPE 165
165   FORMAT('    NEXT EXPERIMENT  (MIN:SEC)       0: 0',
     1        T45,'S  SAMPLE')
      TYPE 166
166   FORMAT(T45,'Z  QUIT')
      HAZA(3)=52
      HAZA(4)=32
      CALL PRINT(HAZA)
      TYPE 190
190   FORMAT( 'PLEASE ENTER YOUR COMMAND',$)
30    FORMAT( )
      IF(ICOM.NE.65) GOTO 50
      TYPE 30
      CALL PRINT(HAZA)
      TYPE 40
40    FORMAT('UNDER AUTOMATIC OPERATION')

C  * * *  CHECK USER INPUT  * * *
50    CALL IPOKE("44,"010100.OR.IPEEK("44))
      ITEMP=ITTINR()
      CALL IPOKE("44,"167677.AND.IPEEK("44))
      IF (ITEMP.GE.0) GOTO 200

C  * * *  CHECK FOR JUMP TO AUTO  * * *
      IF(ICOM.EQ.65.AND.AUTOTN.EQ.0.0) GOTO 207

C  * * *  CHECK FOR TIME AND TEMP UPDATE  * * *
77    CALL GTIM(KTIME)
      CALL CVTTIM(KTIME,KHRS,KMIN,KSEC,KTICKS)
      IF (KSEC.EQ.ISEC) GOTO 50
      ISEC=KSEC

C  * * *  REPRINT TIMES AND TEMPS  * * *
      ENCODE(8,80,PRNTIM) KHRS,KMIN,KSEC
80    FORMAT(I2,':',I2,':',I2)

C  * * *  GET CURRENT TEMPS  * * *
      K=1
      DO 90 I=1,6
      CALL IPOKE("176770,K)
      J=IPEEK("176772)
      IF(J.GT."3777) J=J-"10000
      CVOLT(I)=J/389.905
C  * * * 389.905="3777/5.25 * * *
```

```
      IF(IACTVS(I).EQ.0) GOTO 88
      IF(CVOLT(I).GE.RLALIM.AND.CVOLT(I).LE.RUALIM) GOTO 88
      HAZ(1)=7
      CALL PRINT(HAZ)
88    RBB=FLOAT(IPPRBB(I))/2.0+1000.0
      VRS=119756.88
      RS=22075.0
      TEMP=VRS/(VRS/(RBB+RS)+CVOLT(I)/1000)-RS
C* * * WHERE 119756.88=VB*22075, VB=5.425 VOLTS * * *
      CTEMP(I)=PPA(I)-PPB(I)*ALOG(RB)
      K=K+"400
90    CONTINUE

C  * * *  PRINT TIME  * * *
      TYPE 30
      HAZA(3)=32
      HAZA(4)=38
      CALL PRINT(HAZA)
      CALL PRINT(PRNTIM)

C  * * *  PRINT CURRENT MILLIVOLTS  * * *
      TYPE 30
      HAZA(3)=38
      HAZA(4)=56
      CALL PRINT(HAZA)
      TYPE 95,CVOLT
95    FORMAT(6F8.3)

C  * * *  PRINT CURRENT TEMPS  * * *
      TYPE 30
      HAZA(3)=39
      CALL PRINT(HAZA)
      TYPE 97,CTEMP
97    FORMAT(6F8.3)

C  * * *  UPDATE AND PRINT NEXT AUTO TIME  * * *
      IF(AUTOTN.EQ.0.0) GOTO 99
      IF(IAUTOC.EQ.0) GOTO 96
      TIME=65536.0*KTIM(1)+KTIM(2)
      IF(AUTOTU.GT.TIME) AUTOTU=AUTOTU-5384000.0
      AUTOTN=AUTOTN-(TIME-AUTOTU)
      IF(AUTOTN.LT.0.0) AUTOTN=0.0
      AUTOTU=TIME
96    TYPE 30
      HAZA(3)=50
      HAZA(4)=66
      CALL PRINT(HAZA)
      KTIM(1)=INT(AUTOTN/65536.0)
      KTIM(2)=INT(AUTOTN-65536.0*KTIM(1))
      CALL CVTTIM(KTIME,KHRS,KMIN,KSEC,KTICKS)
      TYPE 98,KMIN,KSEC
98    FORMAT(I2,':',I2)

C  * * *  RESTORE CURSOR  * * *
99    TYPE 30
      HAZA(3)=53
      HAZA(4)=59
      CALL PRINT(HAZA)
      GOTO 50

C  * * *  PERFORM RECIEVED COMMAND  * * *
200   ICOM=ITEMP
205   TYPE 30
207   IF (ICOM.NE.65) GOTO 215
      IF (IAUTOC.NE.0) GOTO 210
```

```
      IAUTOC=1
      CALL GTIM(KTIME)
      AUTOTU=65536.0*KTIM(1)+KTIM(2)
210   IF(AUTOTN.EQ.0.0) CALL AUTO
215   IF (ICOM.EQ.67) CALL CALC
      IF (ICOM.EQ.68) CALL DATA
      IF (ICOM.EQ.90) GOTO 300
      IF (ICOM.EQ.83) CALL SAMP
220   IF (ICOM.GT.68.AND.ICOM.LT.83) CALL UPDATE
      GOTO 5
300   STOP'     END    OF     PROGRAM    '
      END
```

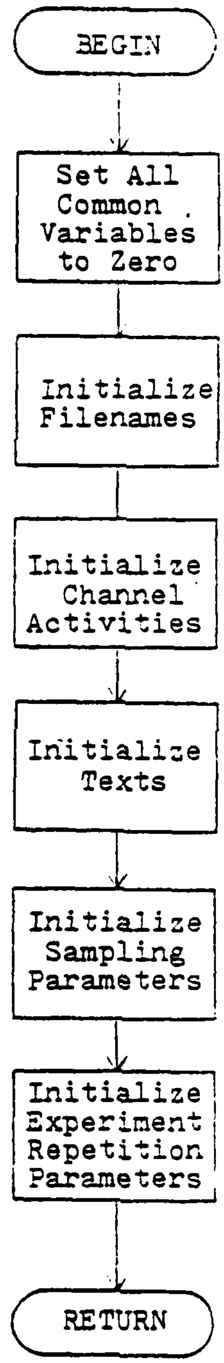


Figure 15. Flowchart of Subroutine INIT

```
      SUBROUTINE INIT
C         THIS SUBROUTINE INITINIALIZES THE SYSTEM PARAMATERS
C   * * *  COMMON VARIABLES   * * *
      LOGICAL*1        HAZ(3),HAZA(5)
      INTEGER          I2(256)
      INTEGER*4        I4(12)
      REAL*4           R4(64)
      INTEGER          IDATA(1800)

      COMMON  /HAZEL/HAZ,HAZA
      COMMON  /VARBLE/I2,I4,R4
      COMMON  /TEMPS/IDATA
C   * * *  END COMMON BLOCK   * * *

C   * * *  BEGIN COMMON DESIGNATIONS   * * *
      INTEGER IACTVS(6),IACTVC(6),IPROBE(6),IHEAT(6),IPPRBB(6)
      INTEGER IFILE(6,3),ITEXT(6,20),ICDATE(6,3)

      EQUIVALENCE     (ICOM,I2(1)),               (IACTVS(1),I2(2))
      EQUIVALENCE     (IPROBE(1),I2(8)),          (IHEAT(1),I2(14))
      EQUIVALENCE     (IPPRBB(1),I2(20)),         (IPSR,I2(26))
      EQUIVALENCE     (IPSP,I2(27)),              (ISR,I2(28))
      EQUIVALENCE     (ISP,I2(29)),               (ITEXT(1,1),I2(30))
      EQUIVALENCE     (ICDATE(1,1),I2(150)),      (IACTVC(1),I2(168))
      EQUIVALENCE     (IFILE(1,1),I2(174)),       (IAUTOC,I2(192))
      EQUIVALENCE     (IEXREP,I2(193))
      EQUIVALENCE     (PPVBB(1),R4(25)),          (CTEMP(1),R4(31))
      EQUIVALENCE     (RLTEMP(1),R4(37)),         (RUALIM,R4(43))
      EQUIVALENCE     (RLALIM,R4(44)),            (AUTOTH,R4(45))
      EQUIVALENCE     (AUTOTU,R4(46)),            (EXINT,R4(47))
C   * * *  END COMON DESIGNATIONS   * * *


C * * *   BEGIN CODE SEGMENT   * * *
      DO 100 I=1,256
      I2(I)=0
100   CONTINUE

      DO 110  I=1,64
      R4(I)=0.0
110   CONTINUE

      DO 120 I=1,12
      I4(I)=0
120   CONTINUE

      DO 5 I=1,6
      IFILE(I,1)='AA'
      IFILE(I,3)='00'
5     CONTINUE
      IFILE(1,2)='A0'
      IFILE(2,2)='B0'
      IFILE(3,2)='C0'
      IFILE(4,2)='D0'
      IFILE(5,2)='E0'
      IFILE(6,2)='F0'
```

```
      DO 10 I=1,6
      IACTVS(I)='ON'
      IACTVC(I)='ON'
      IHEAT(I)=120
10    CONTINUE

      DO 20 J=1,6
      DO 20 I=1,20
      ITEXT(J,I)="52124
20    CONTINUE

      IPSR=10
      IPSP=60
      ISR=10
      ISP=60

      RUALIM=6.0
      RLALIM=-6.0

      EXINT=60.0
      IEXREP=1
      AUTOTN=30.0

      RETURN
      END
```

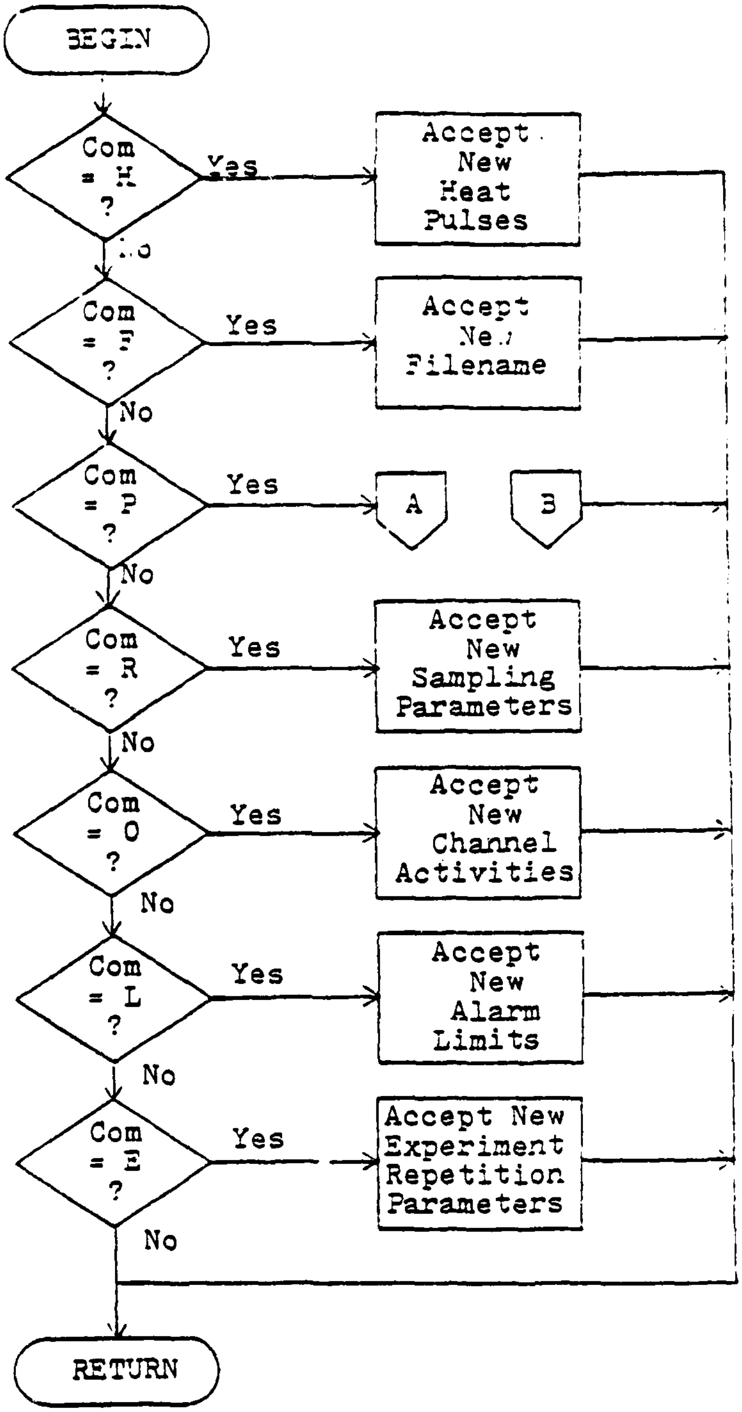


Figure 16. Flowchart of Subroutine UPDATE

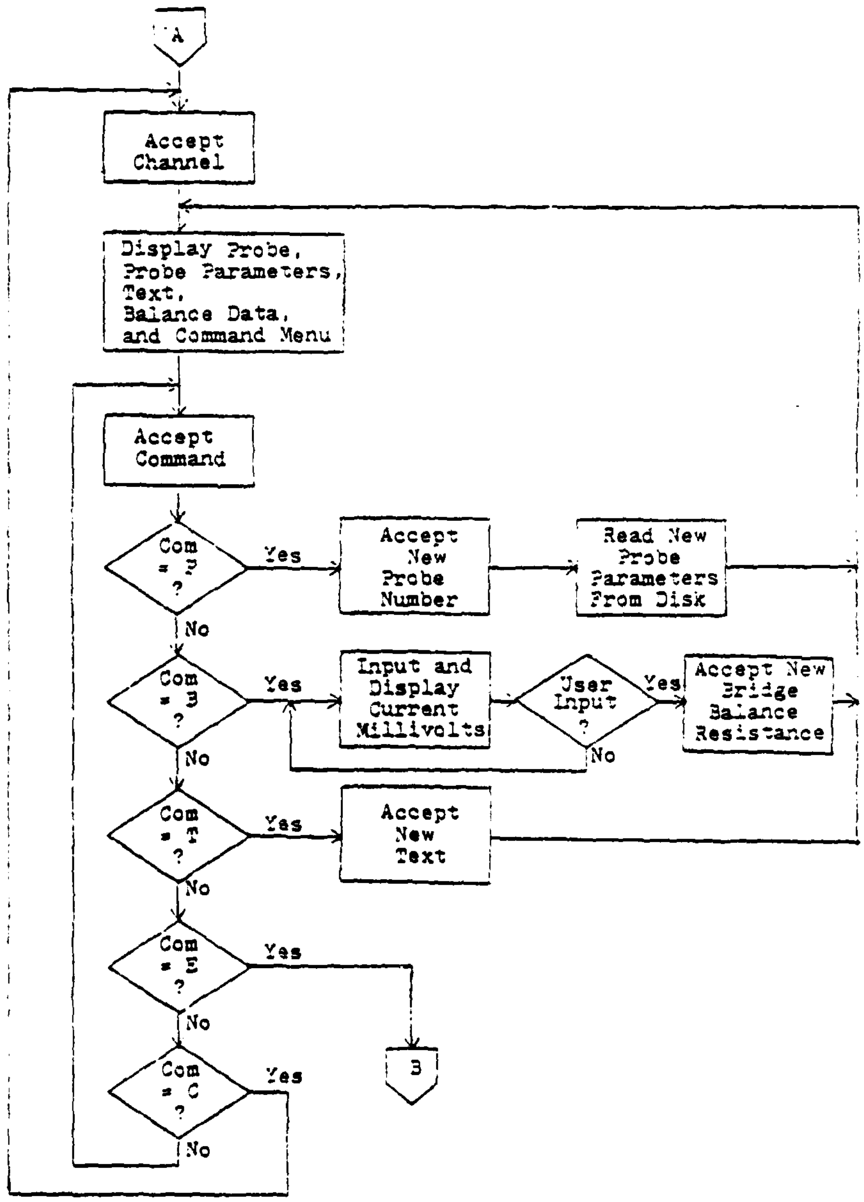


Figure 16. Continued

```
      SUBROUTINE UPDATE

C     THIS SUBROUTINE ALLOWS THE USER TO CHANGE
C     PARAMETERS TO TAILOR SYSTEM CONFIGURATION

C * * * COMMON VARIABLES * * *
      LOGICAL*1      HAZ(3),HAZA(5)
      INTEGER        I2(256)
      INTEGER*4      I4(12)
      REAL*4         R4(64)
      INTEGER        IDATA(1800)

      COMMON  /HAZEL/HAZ,HAZA
      COMMON  /VARBLE/I2,I4,R4
      COMMON  /TEMPS/IDATA
C * * * END COMMON BLOCK * * *

C * * * BEGIN COMMON DESIGNATIONS * * *
      INTEGER IACTVS(6),IACTVC(6),IPROBE(6),IHEAT(6),IPPRBB(6)
      INTEGER IFILE(6,3),ITEXT(6,20),ICDATE(6,3)
      INTEGER*4 IPPTBB(6)
      REAL*4 PPRBB(6),PPBETA(6),PPA(6),PPB(6),PPVBB(6)
      REAL*4 CTEMP(6),RLTEMP(6)

      EQUIVALENCE    (ICOM,I2(1)),          (IACTVS(1),I2(2))
      EQUIVALENCE    (IPROBE(1),I2(8)),     (IHEAT(1),I2(14))
      EQUIVALENCE    (IPPRBB(1),I2(20)),    (IPSR,I2(26))
      EQUIVALENCE    (IPSP,I2(27)),         (ISR,I2(28))
      EQUIVALENCE    (ISP,I2(29)),          (ITEXT(1,1),I2(30))
      EQUIVALENCE    (ICDATE(1,1),I2(150)), (IACTVC(1),I2(168))
      EQUIVALENCE    (IFILE(1,1),I2(174)),  (IAUTOC,I2(192))
      EQUIVALENCE    (IEXREP,I2(193))
      EQUIVALENCE    (IPPTBB(1),I4(1))
      EQUIVALENCE    (PPRBB(1),R4(1)),      (PPBETA(1),R4(7))
      EQUIVALENCE    (PPA(1),R4(13)),       (PPB(1),R4(19))
      EQUIVALENCE    (PPVBB(1),R4(25)),     (CTEMP(1),R4(31))
      EQUIVALENCE    (RLTEMP(1),R4(37)),    (RUALIM,R4(43))
      EQUIVALENCE    (RLALIM,R4(44)),       (AUTOTN,R4(45))
      EQUIVALENCE    (AUTOTU,R4(46)),       (EXINT,R4(47))
C * * * END COMON DESIGNATIONS * * *

C * * * BEGIN LOCAL VARIABLES * * *
      INTEGER JDATA(12)
      EQUIVALENCE(ATEM,JDATA(5)),(BTEM,JDATA(7))
      EQUIVALENCE(RB0TEM,JDATA(9)),(BETAT,JDATA(11))
      EQUIVALENCE(IYEAR,JDATA(4))
      EQUIVALENCE(IMONTH,JDATA(2)),(IDAY,JDATA(3))
C * * * END LOCAL VARIABLES * * *


C * * * BEGIN CODE SEGMENT * * *

C * * * CLEAR ROW AND GET SET TO TYPE REQUEST * * *
      HAZA(3)=52
      HAZA(4)=32
5     CALL PRINT(HAZA)
      HAZ(1)=27
      HAZ(2)=89
      CALL PRINT(HAZ)
20    FORMAT( )


C * * * CHANGE HEAT PULSE DURATIONS * * *
      IF (ICOM.NE.72) GOTO 100
```

```
        TYPE 30
30      FORMAT('UPDATE HEAT PULSE DURATION OF WHICH CHANNEL ?  ',$)
        ACCEPT 40,I
40      FORMAT( I  )
        TYPE 20
        IF (I.LT.1.OR.I.GT.6) GOTO 5
        CALL PRINT(HAZA)
        CALL PRINT(HAZ)
        TYPE 50
50      FORMAT('ENTER NEW HEAT PULSE DURATION   ',$)
        ACCEPT 60,IHEAT(I)
60      FORMAT( I3 )
        GOTO 1000


C  * * *  CHANGE FILENAME  * * *
100     IF(ICOM.NE.70) GOTO 200
        TYPE 110
110     FORMAT( 'UPDATE FILENAME OF WHICH CHANNEL ?    ',$)
        ACCEPT 120,I
120     FORMAT( I  )
        TYPE 20
        IF(I.LT.1.OR.I.GT.6) GOTO 5
        CALL PRINT(HAZA)
        CALL PRINT(HAZ)
        TYPE 130
130     FORMAT('ENTER NEW FILENAME - - - AAANNN ',$)
140     ACCEPT 150,IFILE(I,1),IFILE(I,2),IFILE(I,3)
150     FORMAT(3A2)
        TYPE 20
        DECODE(2,160,IFILE(I,3),ERR=170)IFILEA
160     FORMAT(I2)
        IF(IFILE(I,2).GT."30000.AND.IFILE(I,2).LT."35000) GOTO 1000
170     CALL PRINT(HAZA)
        CALL PRINT(HAZ)
        TYPE 180
180     FORMAT( 'INCORRECT ENTRY - TRY AGAIN - AAANNN   ',$)
        GOTO 140


C  * * *  CHANGE PROBES, TEXTS, AND BRIDGE BALANCES  * * *
200     IF (ICOM.NE.80) GOTO 400
        HAZ(1)=26
        HAZ(2)="200
205     CALL PRINT(HAZ)
        TYPE 20
        TYPE 210
210     FORMAT('UPDATE PROBE INFORMATION OF WHICH CHANNEL ? ',$)
        ACCEPT 215,I
215     FORMAT( I  )
        TYPE 20
        IF (I.LT.1.OR.I.GT.6) GOTO 205
217     CALL PRINT(HAZ)
        TYPE 218,I
218     FORMAT('PROBE IS ON CHANNEL - ',I)
        TYPE 220,IPROBE(I)
220     FORMAT('PROBE NUMBER          - ',I3)
        TYPE 225,PPR80(I)
225     FORMAT('R80                   - ',F8.2)
        TYPE 230,PPBETA(I)
230     FORMAT('BETA                  - ',F8.3)
        TYPE 235,PPA(I)
235     FORMAT('A                     - ',F8.4)
```

```
        TYPE 240,PPB(I)
240     FORMAT('B                        - ',F8.4)
        TYPE 245,(ICDATE(I,K),K=1,3)
245     FORMAT('PROBE CALIBRATED ON - ',I2,'/',I2,'/',I2)
        TYPE 250,IPPRBB(I)
250     FORMAT('BRIDGE BALANCE RESISTANCE - - ',I4)
        TYPE 255,PPVBB(I)
255     FORMAT('BRIDGE BALANCE VOLTAGE    - - ',F5.2)
        CALL CVTTIM(IPPTBB(I),KHRS,KMIN,KSEC,KTICKS)
        TYPE 260,KHRS,KMIN,KSEC
260     FORMAT('BRIDGE BALANCED AT            - - ',I2,':',I2,':',I2)
        TYPE 265,(ITEXT(I,K),K=1,20)
265     FORMAT('TEXT      - - ',20A2)
        TYPE 20
        TYPE 20
        TYPE 270
270     FORMAT('TYPE "P" TO CHANGE PROBE')
        TYPE 275
275     FORMAT('      "B" TO CHANGE BALANCE')
        TYPE 280
280     FORMAT('      "T" TO CHANGE TEXT')
        TYPE 283
283     FORMAT('      "C" TO WORK WITH NEW CHANNEL')
        TYPE 285
285     FORMAT('   OR "E" TO RETURN TO SMORGASBOARD ',$)
288     ACCEPT 290,K
290     FORMAT( A )
        IF (K.EQ.'E') GOTO 1000
C  * * *  CHANGE PROBE NUMBER AND CALIBRATION DATA * * *
        IF (K.NE.'P') GOTO 330
        TYPE 295
295     FORMAT('ENTER NEW PROBE NUMBER ',$)
        CALL ASSIGN(3,'FD0:PROBE.DAT',13)
        READ (3)JNUM
        INUM=JNUM*12
        READ(3)(IDATA(K),K=1,INUM)
        CALL CLOSE(3)
        ACCEPT 298,K
298     FORMAT(I3)
        J=1
300     IF(K.EQ.IDATA(J)) GOTO 310
        J=J+12
        IF(J.LT.INUM) GOTO 300
        TYPE 305
305     FORMAT('NO PROBE EXISTS WITH THAT DESIGNATION')
        GOTO 217
310     IPROBE(I)=K
        DO 320 K=1,12
        JDATA(K)=IDATA(J)
        J=J+1
320     CONTINUE
        PPA(I)=ATEM
        PPB(I)=BTEM
        PPRB0(I)=RB0TEM
        PPBETA(I)=BETAT
        ICDATE(I,1)=IMONTH
        ICDATE(I,2)=IDAY
        ICDATE(I,3)=IYEAR
        GOTO 217
C  * * *  CHANGE BRIDGE BALANCE   * * *
330     IF (K.NE.'B') GOTO 370
        TYPE 335
335     FORMAT('CURRENT RAW DATA       CURRENT MILLIVOLTS')
        K=1
        IF(I.EQ.1) GOTO 345
```

```
      DO 340 J=1,I-1
      K=K+"400
340   CONTINUE
345   HAZA(3)=51
      HAZA(4)=32
347   TYPE 20
      CALL PRINT(HAZA)
      ISEC=KSEC
      CALL IPOKE("176770,K)
      J=IPEEK("176772)
      J1=J
      IF(J.GT."3777) J1=J-"10000
      PPV88(I)=J1/389.9047619
      TYPE 350,J,PPV88(I)
350   FORMAT(5X,O6,10X,F7.4)
      TYPE 355
355   FORMAT('PRESS ANY KEY WHEN DESIRED BALANCE IS REACHED ')
360   CALL IPOKE("44,"010100.OR.IPEEK("44))
      J=ITTINR()
      CALL IPOKE("44,"167677.AND.IPEEK("44))
      IF(J.GE.0) GOTO 365
      CALL GTIM(IPPTB8(I))
      CALL CVTTIM(IPPTB8(I),KHRS,KMIN,KSEC,KTICKS)
      IF(KSEC.EQ.ISEC) GOTO 360
      GOTO 347
365   TYPE 366
366   FORMAT('ENTER BRIDGE RESISTANCE BALANCE ',$)
      ACCEPT 367,IPPR88(I)
367   FORMAT(I4)
      GOTO 217
C  * * *  CHANGE TEXT  * * *
370   IF(K.NE.'T') GOTO 385
      TYPE 375
375   FORMAT('ENTER NEW TEXT - 12345678901234567890',
     1        '12345678901234567890')
      TYPE 380
380   FORMAT('                    ',$)
      ACCEPT 382,(ITEXT(I,K),K=1,20)
382   FORMAT(20A2)
      GOTO 217
385   IF(K.EQ.'C') GOTO 205
      TYPE 390
390   FORMAT('INVALID ENTRY - - TRY AGAIN',$)
      GOTO 200


C  * * *  CHANGE SAMPLE PARAMETERS  * * *
400   IF(ICOM.NE.82) GOTO 500
      TYPE 410
410   FORMAT('ENTER NEW VALUE OF PRESAMPLE DURATION (SEC)  ',$)
      ACCEPT 420,IPSR
420   FORMAT(I4)
      TYPE 20
      CALL PRINT(HAZA)
      CALL PRINT(HAZ)
      TYPE 430
430   FORMAT('ENTER NEW PRESAMPLE FREQUENCY (HZ)  ',$)
      ACCEPT 420,IPSP
      IPSP=60/IPSP
      IPSR=IPSR*60/IPSP
      TYPE 20
      CALL PRINT(HAZA)
      CALL PRINT(HAZ)
      TYPE 440
```

```
440     FORMAT('ENTER NEW VALUE OF SAMPLE DURATION (SEC)  ',$)
        ACCEPT 420,ISR
        TYPE 20
        CALL PRINT(HAZA)
        CALL PRINT(HAZ)
        TYPE 450
450     FORMAT('ENTER NEW SAMPLE FREQUENCY (HZ)  ',$)
        ACCEPT 420,ISP
        ISP=60/ISP
        ISR=ISR*60/ISP
        GOTO 1000


C  * * *  CHANGE CHANNEL ACTIVITIES  * * *
500     IF(ICOM.NE.79) GOTO 600
        TYPE 510
510     FORMAT('ENTER SAMPLE CHANNEL ACTIVITY AS A SIX BIT BINARY',
     1         ' NUMBER  ',$)
        ACCEPT 520,IACTVS
520     FORMAT(6O1)
        TYPE 20
        CALL PRINT(HAZA)
        CALL PRINT(HAZ)
        TYPE 540
540     FORMAT('ENTER CALCULATE CHANNEL ACTIVITY AS A SIX BIT',
     1         ' BINARY NUMBER  ',$)
        ACCEPT 550,IACTVC
550     FORMAT(6O1)
        DO 570 J=1,6
        IF(IACTVS(J).EQ.0) GOTO 560
        IACTVS(J)='ON'
560     IF(IACTVC(J).EQ.0) GOTO 570
        IACTVC(J)='ON'
570     CONTINUE


C * * *  CHANGE ALARM LIMITS  * * *
600     IF(ICOM.NE.76) GOTO 700
        TYPE 610
610     FORMAT('ENTER NEW UPPER ALARM LIMIT (MV) ',$)
        ACCEPT 620,RUALIM
620     FORMAT(F5.2)
        TYPE 20
        CALL PRINT(HAZA)
        CALL PRINT(HAZ)
        TYPE 630
630     FORMAT('ENTER NEW LOWER ALARM LIMIT (MV) ',$)
        ACCEPT 640,RLALIM
640     FORMAT(F5.2)
        GOTO 1000


C  * * *  CHANGE EXPERIMENT REPEAT DATA  * * *
700     IF(ICOM.NE.69) GOTO 800
        TYPE 710
710     FORMAT('ENTER NEW EXPERIMENT INTERVAL (SEC) ',$)
        ACCEPT 720,J
720     FORMAT(I4)
        EXINT=J*60.0
       TYPE 20
        CALL PRINT(HAZA)
        CALL PRINT(HAZ)
```

```
        TYPE 730
730     FORMAT('ENTER NEW EXPERIMENT REPETITIONS ',$)
        ACCEPT 740,IEXREP
740     FORMAT(I3)
        TYPE 20
        CALL PRINT(HAZA)
        CALL PRINT(HAZ)
        TYPE 750
750     FORMAT('ENTER NEW TIME UNTL NEXT EXPERIMENT (SEC) ',$)
        ACCEPT 760,J
760     FORMAT(I4)
        AUTOTN=J*60
        IAUTOC=0
        GOTO 1000


300     CONTINUE
1000    HAZ(2)="200
        RETURN
        END
```

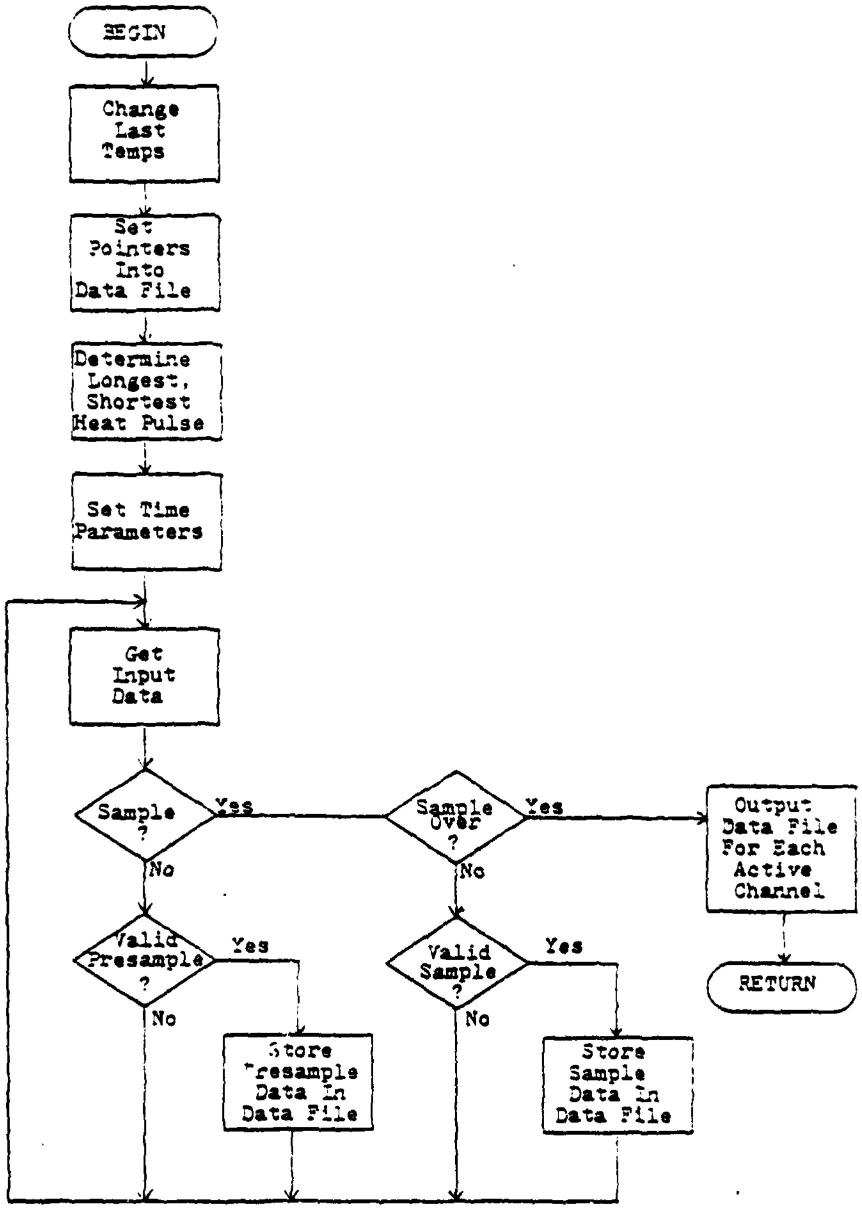


Figure 17. Flowchart of Subroutine SAMP

```
      SUBROUTINE SAMP
C     THIS SUBROUTINE PERFORMS THE ACTUAL SAMPLING OF THE
C     DATA AND CREATES DATA FILES ON DISK FOR EACH ACTIVE
C     CHANNEL
C .* * *  COMMON VARIABLES  * * *
      LOGICAL*1       HAZ(3),HAZA(5)
      INTEGER         I2(256)
      INTEGER*4       I4(12)
      REAL*4          R4(64)
      INTEGER         IDATA(1800)

      COMMON  /HAZEL/HAZ,HAZA
      COMMON  /VARBLE/I2,I4,R4
      COMMON  /TEMPS/IDATA
C   * * *  END COMMON BLOCK  * * *

C   * * *  BEGIN COMMON DESIGNATIONS  * * *
      INTEGER IACTVS(6),IACTVC(6),IPROBE(6),IHEAT(6),IPPRBB(6)
      INTEGER IFILE(6,3),ITEXT(6,20),ICDATE(6,3)
      INTEGER*4 IPPTBB(6)
      REAL*4 PPRBB(6),PPBETA(6),PPA(6),PPB(6),PPVBB(6)
      REAL*4 CTEMP(6),RLTEMP(6)

      EQUIVALENCE    (ICOM,I2(1)),              (IACTVS(1),I2(2))
     EQUIVALENCE    (IPROBE(1),I2(8)),          (IHEAT(1),I2(14))
      EQUIVALENCE    (IPPRBB(1),I2(20)),        (IPSR,I2(26))
      EQUIVALENCE    (IPSP,I2(27)),             (ISR,I2(28))
      EQUIVALENCE    (ISP,I2(29)),              (ITEXT(1,1),I2(30))
      EQUIVALENCE    (ICDATE(1,1),I2(150)),     (IACTVC(1),I2(168))
     EQUIVALENCE    (IFILE(1,1),I2(174)),       (IAUTOC,I2(192))
      EQUIVALENCE    (IPPTBB(1),I4(1))
      EQUIVALENCE    (PPRBB(1),R4(1)),          (PPBETA(1),R4(7))
      EQUIVALENCE    (PPA(1),R4(13)),           (PPB(1),R4(19))
      EQUIVALENCE    (PPVBB(1),R4(25)),         (CTEMP(1),R4(31))
      EQUIVALENCE    (RLTEMP(1),R4(37)),        (RUALIM,R4(43))
C   * * *  END COMON DESIGNATIONS  * * *

C   * * * BEGIN LOCAL DECLARATIONS  * * *
      INTEGER IOFFST(6)
      INTEGER ITIMC(6)
      INTEGER*2 KTIM(2)
      INTEGER ITIMB(6)
      INTEGER IFILET(18)
      INTEGER IFILEA(7)
      INTEGER KDATE(3)
      INTEGER*4 JTIM,ISTIM
      EQUIVALENCE (KMON,KDATE(1)),(KDAY,KDATE(2))
      EQUIVALENCE (KYEAR,KDATE(3)),(JTIM,KTIM(1))
C   * * *  END LOCAL DECLARATIONS  * * *

C   * * *  BEGIN CODE SEGMENT  * * *
      HAZ(1)=26
      TYPE 1
1     FORMAT('NOW PERFORMING SAMPLE')
      CALL PRINT(HAZ)

C   * * *  CHANGE LAST TEMPERATURES  * * *
      DO 10 I=1,6
      RLTEMP(I)=CTEMP(I)
10    CONTINUE

C   * * *  CLEAR IDATA  * * *
      DO 5 I=1,1800
```

```
      IDATA(I)=1
5     CONTINUE
C   * * *  SET POINTERS INTO IDATA  * * *
      JTEMP=2+IPSR+ISR
      K=1
      DO 20 I=1,6
      IOFFST(I)=IPSR+2
      IF (IACTVS(I).EQ.0) GOTO 20
      IOFFST(I)=K
      K=K+JTEMP
20    CONTINUE
      J1=IOFFST(1)
      J2=IOFFST(2)
      J3=IOFFST(3)
      J4=IOFFST(4)
      J5=IOFFST(5)
      J6=IOFFST(6)
C   * * *  FIGURE OUT SHORTEST AND LONGEST HEATING PULSES  * * *
      IHEATS=1200
      DO 60 I=1,6
      IF (IACTVS(I).EQ.0) GOTO 60
      IF (IHEATS.LE.IHEAT(I)) GOTO 60
      IHEATS=IHEAT(I)
60    CONTINUE
      IHEATL=0
      DO 70 I=1,6
      IF (IACTVS(I).EQ.0) GOTO 70
      IF (IHEATL.GE.IHEAT(I)) GOTO 70
      IHEATL=IHEAT(I)
70    CONTINUE
C   * * *  SET TIMES TO PERFORM PRESAMPLE  * * *
      ITIMA=1
      ITIME=IPSR*IPSP+ITIMA+IHEATL
      ITIMD=ITIME-IHEATS
      DO 75 I=1,6
      ITIMC(I)=ITIME
      ITIMB(I)=ITIME
75    CONTINUE
      DO 80 I=1,6
      IF (IACTVS(I).EQ.0) GOTO 80
      ITIMC(I)=ITIME-IHEAT(I)
      ITIMB(I)=ITIMC(I)-IPSR*IPSP
80    CONTINUE
      ITIMF=ITIME+ISR*ISP
      ICOUNT=1
C   * * *  CLEAR OUTPUT VARIABLE  * * *
      IPIO=0
C   * * *  SET TIME OFFSET  * * *
      CALL GTIM(JTIM)
      ITOFF=KTIM(2)
C   * * *  PERFORM EXPERIMENT  * * *
90    CALL GTIM(JTIM)
      ITIM=KTIM(2)-ITOFF
      IF(ITIM.NE.ITIMA) GOTO 90
      ITIMA=ITIMA+1
      CALL IPOKE("176770,"1)
      IDATA(J1)=IPEEK("176772)
      CALL IPOKE("176770,"401)
      IDATA(J2)=IPEEK("176772)
```

```
      CALL IPOKE("176770,"1001)
      IDATA(J3)=IPEEK("176772)
      CALL IPOKE("176770,"1401)
      IDATA(J4)=IPEEK("176772)
      CALL IPOKE("176770,"2001)
      IDATA(J5)=IPEEK("176772)
      CALL IPOKE("176770,"2401)
      IDATA(J6)=IPEEK("176772)

      IF(ITIM.GE.ITIME) GOTO 220

      IF(ITIM.NE.ITIMB(1)) GOTO 110
      IF(ITIM.EQ.ITIMC(1)) GOTO 100
      IPIO=IPIO.OR."1
      J1=J1+1
      ITIMB(1)=ITIMB(1)+IPSP
      GOTO 110
100   IPIO="177776.AND.IPIO.OR."400

110   IF(ITIM.NE.ITIMB(2)) GOTO 130
      IF(ITIM.EQ.ITIMC(2)) GOTO 120
      IPIO=IPIO.OR."2
      J2=J2+1
      ITIMB(2)=ITIMB(2)+IPSP
      GOTO 130
120   IPIO="177775.AND.IPIO.OR."1000

130   IF(ITIM.NE.ITIMB(3)) GOTO 150
      IF(ITIM.EQ.ITIMC(3)) GOTO 140
      IPIO=IPIO.OR."4
      J3=J3+1
      ITIMB(3)=ITIMB(3)+IPSP
      GOTO 150
140   IPIO="177773.AND.IPIO.OR."2000

150   IF(ITIM.NE.ITIMB(4)) GOTO 170
      IF(ITIM.EQ.ITIMC(4)) GOTO 160
      IPIO=IPIO.OR."10
      J4=J4+1
      ITIMB(4)=ITIMB(4)+IPSP
      GOTO 170
160   IPIO="177767.AND.IPIO.OR."4000

170   IF(ITIM.NE.ITIMB(5)) GOTO 190
      IF(ITIM.EQ.ITIMC(5)) GOTO 180
      IPIO=IPIO.OR."20
      J5=J5+1
      ITIMB(5)=ITIMB(5)+IPSP
      GOTO 190
180   IPIO="177757.AND.IPIO.OR."10000

190   IF(ITIM.NE.ITIMB(6)) GOTO 210
      IF(ITIM.EQ.ITIMC(6)) GOTO 200
      IPIO=IPIO.OR."40
      J6=J6+1
      ITIMB(6)=ITIMB(6)+IPSP
      GOTO 210
200   IPIO="177737.AND.IPIO.OR."20000

210   CALL IPOKE("177556,IPIO)
      GOTO 90

220   CALL IPOKE("177556,"100)
      IF (ITIM.EQ.ITIMF) GOTO 300
      IF(ITIM.EQ.ITIME) CALL GTIM(ISTIM)
```

```
      ICOUNT=ICOUNT-1
      IF(ICOUNT.NE.0) GOTO 90
      ICOUNT=ISP
      J1=J1+1
      J2=J2+1
      J3=J3+1
      J4=J4+1
      J5=J5+1
      J6=J6+1
      GOTO 90

C  * * *  CLEAR OUTPUT LIGHTS  * * *
300   CALL IPOKE("177556,0)

C  * * *  OUTPUT TEMORARY PRESAMPLE DATA  * * *
      TYPE 310
310   FORMAT('EXPERIMENT IS DONE')
      TYPE 320
320   FORMAT('FIRST TEN PRESAMPLE POINTS OF EACH CHANNEL')
      DO 345 I=1,6
      TYPE 340,(IDATA(J),J=IOFFST(I),IOFFST(I)+9)
340   FORMAT(10O7)
345   CONTINUE
      TYPE 410
410   FORMAT( )
      TYPE 420,(IOFFST(I),I=1,6)
420   FORMAT(6I8)

C  * * *  OUTPUT TEMPORARY SAMPLE DATA  * * *
      TYPE 410
      TYPE 550
550   FORMAT('FIRST TEN SAMPLE POINTS OF EACH CHANNEL')
      TYPE 410
      DO 570 I=1,6
      TYPE 560,(IDATA(J),J=IOFFST(I)+IPSR,IOFFST(I)+IPSR+9)
560   FORMAT(10O7)
570   CONTINUE

C  * * *  OUTPUT DATA FILES  * * *
      IFILEA(1)='FD'
      IFILEA(2)='1:'
      IFILEA(6)='.D'
      IFILEA(7)='AT'
      CALL IDATE(KMON,KDAY,KYEAR)
      DO 590 I=1,18
      IFILET(I)="20040
590   CONTINUE
      DO 610 I=1,6
      IF (IACTVS(I).EQ.0) GOTO 610
      DO 600 J=1,3
      K=I*3-3+J
      IFILET(K)=IFILE(I,J)
600   CONTINUE
610   CONTINUE
      DO 700 I=1,6
      IF (IACTVS(I).EQ.0) GOTO 700
      DO 620 J=1,3
      IFILEA(J+2)=IFILE(I,J)
620   CONTINUE
      CALL ASSIGN(3,IFILEA,14)
      WRITE(3)1,ISTIM,KDATE,I,IFILET,(ITEXT(I,K),K=1,20),
     1       IPROBE(I),PPRB0(I),PPBETA(I),PPA(I),PPB(I),
     4       (ICDATE(I,K),K=1,3),
     2       IPPRBB(I),IPPTBB(I),PPVBB(I),IHEAT(I),IPSR,IPSP,
     3       ISR,ISP,(IDATA(K),K=IOFFST(I),IOFFST(I)+IPSR-ISR-1)
```

```
      CALL CLOSE(3)
700   CONTINUE

C  * * *  WAIT FOR RESPONSE BEFORE RETURNING TO MAIN  * * *
      IF(ICOM.EQ.65) GOTO 920
      TYPE 900
900   FORMAT('PRESS RETURN TO EXIT BACK TO SMORGASBORD')
      ACCEPT 910,I
910   FORMAT( A )
920   RETURN
      END
```

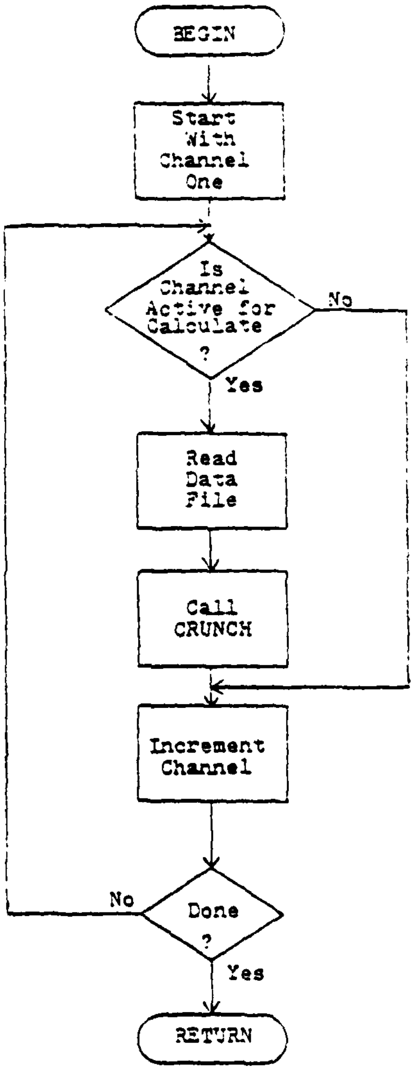


Figure 18. Flowchart of Subroutine CALC

```
      SUBROUTINE CALC
C     THIS SUBROUTIN READS IN THE DESIRED DATA FILES
C     AND CALLS CRUNCH WHICH PERFORMS THE
C     ACTUAL CALCULATIONS

C  * * *  COMMON VARIABLES  * * *
      LOGICAL*1       HAZ(3),HAZA(5)
      INTEGER         I2(256)
      INTEGER*4       I4(12)
      REAL*4          R4(64)
      INTEGER         IDATA(1300)

      COMMON  /HAZEL/HAZ,HAZA
      COMMON  /VARBLE/I2,I4,R4
      COMMON  /TEMPS/IDATA
C  * * *  END COMMON BLOCK  * * *

C  * * *  BEGIN COMMON DESIGNATIONS  * * *
      INTEGER IACTVS(6),IACTVC(6),IPROBE(6),IHEAT(6),IPPRB9(6)
      INTEGER IFILE(6,3),ITEXT(6,20),ICDATE(6,3)

      EQUIVALENCE    (ICDATE(1,1),I2(150)),  (IACTVC(1),I2(168))
      EQUIVALENCE    (IFILE(1,1),I2(174)),   (IAUTOC,I2(192))
C  * * *  END COMON DESIGNATIONS  * * *

C  * * *  BEGIN LOCAL VARIABLES  * * *
      INTEGER IFILEA(7)
C  * * *  END LOCAL VARIABLES  * * *

C  * * *  BEGIN CODE SEGMENT  * * *
      IFILEA(1)='FD'
      IFILEA(2)='1:'
      IFILEA(6)='.D'
      IFILEA(7)='AT'

      HAZ(1)=26
      CALL PRINT(HAZ)

      DO 100 I=1,6
      IF(IACTVC(I).EQ.0) GOTO 100
      TYPE 5,I
5     FORMAT('CRUNCH CHANNEL NUMBER ',I)


      DO 10 J=1,3
      IFILEA(J+2)=IFILE(I,J)
10    CONTINUE

      CALL ASSIGN(3,IFILEA,14)
      READ(3,ERR=60) IDATA
60    CALL CLOSE(3)

      CALL CRUNCH

      TYPE 70
70    FORMAT('BACK FROM CRUNCH')

100   CONTINUE
      ACCEPT 30
30    FORMAT( A )
      RETURN
      END
```

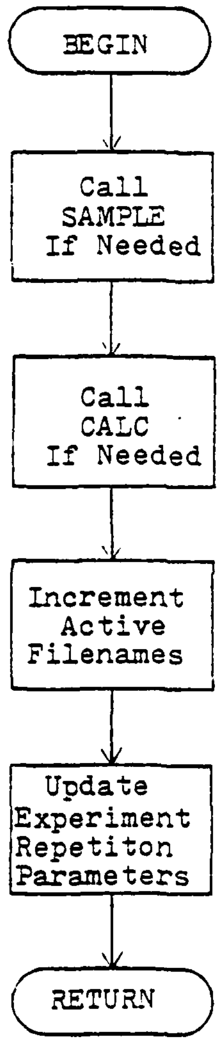


Figure 19. Flowchart of Subroutine AUTO

```
      SUBROUTINE AUTO
C     THIS SUBROUTINE CALL SAMPLE AND CALCULATE,
C     INCREMENTS FILENAMES AND ADJUSTS
C     REPETITION PARAMETERS

C * * *  COMMON VARIABLES  * * *
      LOGICAL*1        HAZ(3),HAZA(5)
      INTEGER          I2(256)
      INTEGER*4        I4(12)
      REAL*4           R4(64)
      INTEGER          IDATA(1800)

      COMMON   /HAZEL/HAZ,HAZA
      COMMON   /VARBLE/I2,I4,R4
      COMMON   /TEMPS/IDATA
C * * *  END COMMON BLOCK  * * *

C * * *  BEGIN COMMON DESIGNATIONS  * * *
      INTEGER IACTVS(6),IACTVC(6),IPROBE(6),IHEAT(6),IPPRBB(6)
      INTEGER IFILE(6,3),ITEXT(6,20),ICDATE(6,3)

      EQUIVALENCE     (ICOM,I2(1)),            (IACTVS(1),I2(2))
      EQUIVALENCE     (ICDATE(1,1),I2(150)),   (IACTVC(1),I2(168))
      EQUIVALENCE     (IFILE(1,1),I2(174)),    (IAUTOC,I2(192))
      EQUIVALENCE     (IEXREP,I2(193))
      EQUIVALENCE     (RLALIM,R4(44)),         (AUTOTN,R4(45))
      EQUIVALENCE     (AUTOTU,R4(46)),         (EXINT,R4(47))
C * * *  END COMON DESIGNATIONS  * * *

C * * *  BEGIN LOCAL VARIABLES * * *
      INTEGER*4 KTIME
      INTEGER   KTIM(2)

      EQUIVALENCE (KTIME,KTIM(1))
C * * *  END LOCAL VARIABLES  * * *

C * * *  BEGIN CODE SEGMENT  * * *
      CALL GTIM(KTIME)
      AUTOTU=65536.0*KTIM(1)+KTIM(2)
      AUTOTN=EXINT

C * * *  PERFORM SAMPLE (IF NEEDED)  * * *
      J=0
      DO 100 I=1,6
      IF(IACTVS(I).NE.0) J=J+1
100   CONTINUE
      IF(J.NE.0) CALL SAMP

C * * *  PERFORM CALCULATE (IF NEEDED)  * * *
      J=0
      DO 110 I=1,6
      IF (IACTVC(I).NE.0) J=J+1
110   CONTINUE
      IF (J.NE.0) CALL CALC

C * * *  INCREMENT FILENAMES  * * *
      DO 180 I=1,6
      IF(IACTVS(I).EQ.0.AND.IACTVC(I).EQ.0) GOTO 180
      DECODE(2,120,IFILE(I,3))IFILEA
120   FORMAT(I2)
      IFILEA=IFILEA+1
      TYPE 125,I,IFILE(I,3),IFILEA
125   FORMAT(I,'          ',A2,'          ',I3)
      IF(IFILEA.LT.100) GOTO 160
```

```
      IFILEA=0
C   * * *  ADD 100   * * *
      IFILEB=IFILE(I,2)+"400
      TYPE 140,IFILE(I,2),IFILEB,IFILEB
140   FORMAT(A2,'        ',A2,'        ',O7)
C   * * *  CHECK GE TO XXX900  * * *
C   * * *  LEAVE AT XXX899  * * *
      IF(IFILEB.GE."34400) GOTO 180
      IFILE(I,2)=IFILEB
150   ENCODE(2,170,IFILE(I,3))IFILEA
170   FORMAT(I2)
      IF(IFILEA.LT.10) IFILE(I,3)=IFILE(I,3).OR."60
      IF(IFILEA.EQ.0) IFILE(I,3)="30060
180   CONTINUE

C   * * *  UPDATE AUTO PARAMETERS  * * *
      IEXREP=IEXREP-1
      IF(IEXREP.NE.0) GOTO 200
      ICOM='B'
      AUTOTN=0.0
      IEXREP=1

200   RETURN
      END
```

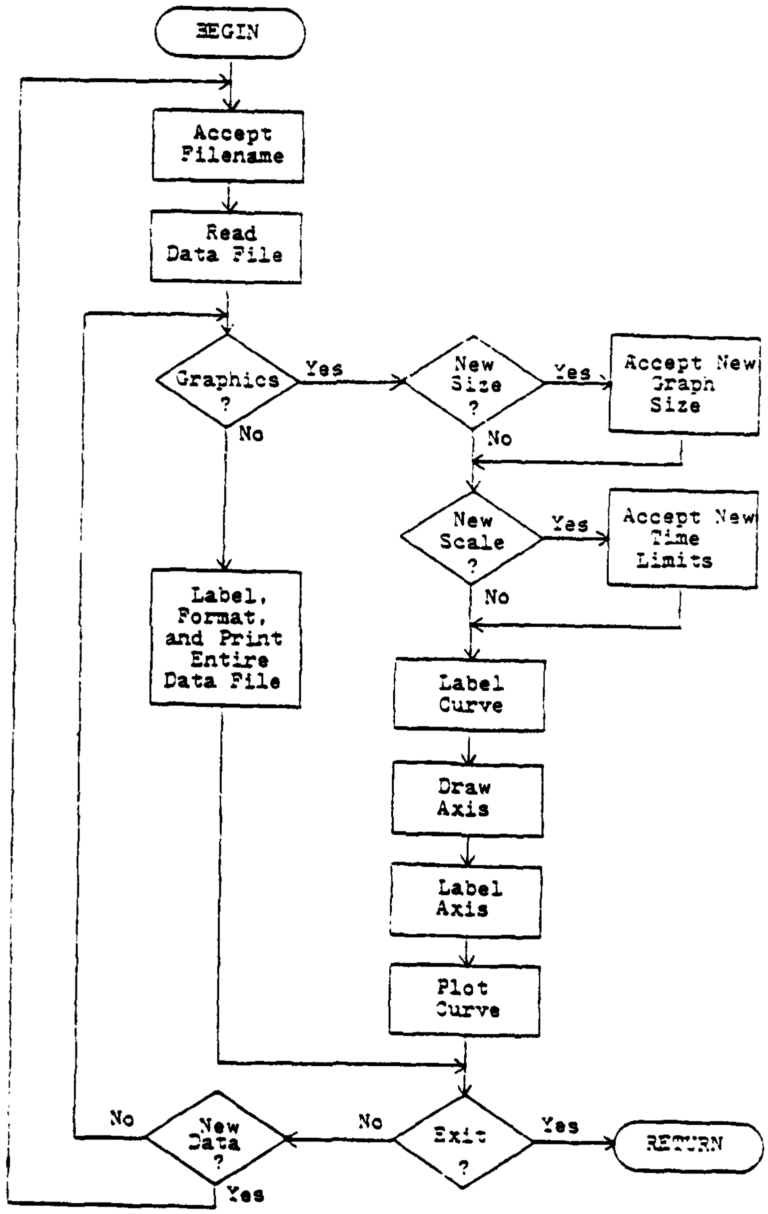


Figure 20. Flowchart of Subroutine DATA

```
      SUBROUTINE DATA
C     THIS SUBROUTINE ALLOWS THE USER TO EITHER LIST
C     AN ENTIRE DATA FILE OR PLOT A FILE ON THE
C     GRAPHICS TERMINAL

C  * * *  COMMON VARIABLES  * * *
      LOGICAL*1        HAZ(3),HAZA(5)
      INTEGER          I2(256)
      INTEGER*4        I4(12)
      REAL*4           R4(64)
      INTEGER          IDATA(1800)

      COMMON  /HAZEL/HAZ,HAZA
      COMMON  /VARBLE/I2,I4,R4
      COMMON  /TEMPS/IDATA
C  * * *  END COMMON BLOCK  * * *


C  * * *  BEGIN LOCAL DESIGNATIONS  * * *
      INTEGER*4 ISTIM
      INTEGER KDATE(3),IFILET(18),ITEXT(20)
      INTEGER ICDATE(3)
      REAL*4 PPRB0,PPBETA,PPA,PPB,PPVBB
      INTEGER*4 IPPTBB
      REAL*4 DATA(1800)

      INTEGER IFILEA(7)
      COMMON  /GRID/IX,IY
C  * * *  END LOCAL DECLARATIONS  * * *

      IFILEA(1)='FO'
      IFILEA(2)='1:'
      IFILEA(6)='.D'
      IFILEA(7)='AT'
10    JTEMP=ITTOUR(26)
      TYPE 30
30    FORMAT( )
      TYPE 40
40    FORMAT( 'ENTER FILENAME YOU WISH TO OPEN  ',$)
      ACCEPT 50,IFILEA(3),IFILEA(4),IFILEA(5)
50    FORMAT( A2,A2,A2 )
      CALL ASSIGN(3,IFILEA,14)
      READ (3,ERR=60) J,ISTIM,KDATE,I,IFILET,ITEXT,IPROBE,
     1       PPRB0,PPBETA,PPA,PPB,ICDATE,IPPRBB,IPPTBB,PPVBB,
     2       IHEAT,IPSR,IPSP,ISR,ISP,IDATA
60    CALL CLOSE(3)
      ICHAN=I

65    TYPE 70
70    FORMAT( 'ENTER "G" FOR GRAPHICS OR "L" FOR LISTING')
      ACCEPT 80,I
80    FORMAT(A)
      IF(I.EQ.'G') GOTO 500

C  * * *  LISTING PACKAGE  * * *
      I=ICHAN
      TYPE 100,J
100   FORMAT('DATA VERSION - -  ',I)
      CALL CVTTIM(ISTIM,KHRS,KMIN,KSEC,KTICKS)
      TYPE 110,KHRS,KMIN,KSEC,KDATE
110   FORMAT( 'SAMPLE TIME  - -  ',I2,':',I2,':',I2,
     1   '                DATE  - -  ',I2,'/',I2,'/',I2 )
      TYPE 120,I
120   FORMAT('CHANNEL DATA RECIEVED FROM - -  ',I)
```

```
        TYPE 130
130     FORMAT('ACTIVE FILES AT SAMPLE TIME')
        TYPE 140,IFILET
140     FORMAT('1- ',3A2,' 2- ',3A2,' 3- ',3A2,' 4- ',3A2,' 5- ',
     1       3A2,' 6- ',3A2)
        TYPE 150,ITEXT
150     FORMAT('TEXT - - ',20A2)
        TYPE 155,IPROBE
155     FORMAT('PROBE DESIGNATION - - ',I3)
        TYPE 160,PPR80,PPBETA,PPA,PPB
160     FORMAT('R80 - - ',F7.2,'  BETA - - ',F8.3,
     1       '  A - - ',F8.4,'  B - - ',F8.4)
        TYPE 165,ICDATE
165     FORMAT('CALIBRATED ON - ',I2,'/',I2,'/',I2)
        CALL CVTTIM(IPPTBB,KHRS,KMIN,KSEC,KTICKS)
        TYPE 170,IPPRBB,PPVBB,KHRS,KMIN,KSEC
170     FORMAT('BRIDGE BALANCE : OHMS - - ',I6,' VOLTS - - ',F5.2,
     1       ' TIME - - ',I2,':',I2,':',I2)
        TYPE 180,IHEAT
180     FORMAT('HEAT PULSE DURATION - - ',I4)
        TYPE 190,IPSR,IPSP
190     FORMAT('PRESAMPLE READINGS - - ',I4,
     1       'PRESAMPLE PERIOD - - ',I3)
        TYPE 200,ISR,ISP
200     FORMAT('    SAMPLE READINGS - - ',I4,
     1       '    SAMPLE PERIOD - - ',I3)
        PPR80=5.25/"3777
        DO 205 K=1,IPSR+ISR
        IF(IDATA(K).GT."3777) IDATA(K)=IDATA(K)-"10000
        DATA(K)=IDATA(K)*PPR80
205     CONTINUE
        TYPE 210
210     FORMAT('PRESAMPLE DATA :     (MILLIVOLTS)')
        DO 225 I=1,IPSR,10
        J=I+9
        IF(IPSR-I.LT.9) J=IPSR
        TYPE 220,(DATA(K),K=I,J)
220     FORMAT(10F7.4)
225     CONTINUE
        TYPE 230
230     FORMAT('SAMPLE DATA :      (MILLIVOLTS)')
        DO 245 I=IPSR+1,IPSR+ISR,10
        J=I+9
        IF(IPSR+ISR-I.LT.9) J=IPSR+ISR
        TYPE 240,(DATA(K),K=I,J)
240     FORMAT(10F7.4)
245     CONTINUE
        GOTO 290

C   * * *  GRAPHICS PACKAGE  * * *
500     JTEMP=ITTOUR(26)

C   * * *  SET GRAPH SIZE
        JTEMP=ITTOUR(13)
        IXMIN=35
        IXMAX=1010
        IYMIN=40
        IYMAX=740
        TYPE 510
510     FORMAT('DO YOU WANT DIFFERENT SIZE GRAPH?')
        ACCEPT 520,I
520     FORMAT(A)
        IF(I.NE.'Y') GOTO 545
        TYPE 530
530     FORMAT('CONSIDER THE SCREEN TO BE 100 BY 100')
```

```
932     TYPE 935
935     FORMAT('ENTER LEFT HORIZONTAL MARGIN FROM 0 TO 99')
        ACCEPT 940,I
940     FORMAT(I2)
        TYPE 945,I
945     FORMAT('ENTER RIGHT HORIZONTAL MARGIN FROM ',I2,' TO 99')
        ACCEPT 950,J
950     FORMAT(I2)
        IF(J.GT.I)GOTO 960
        TYPE 955
955     FORMAT('INCORRECT VALUES - - TRY AGAIN')
        GOTO 932
960     I=(IXMAX-IXMIN)*I/99+IXMIN
        IXMAX=INT((IXMAX-IXMIN)/99.0*J)+IXMIN
        IXMIN=I

962     TYPE 965
965     FORMAT('ENTER LOWER VERTICAL LIMIT FROM 0 TO 99')
        ACCEPT 970,I
970     FORMAT(I2)
        TYPE 975,I
975     FORMAT('ENTER UPPER VERTICAL LIMIT FROM ',I2,' TO 99')
        ACCEPT 980,J
980     FORMAT(I2)
        IF(J.GT.I)GOTO 990
        TYPE 985
985     FORMAT('INCORRECT VERTICAL LIMITS - - TRY AGAIN')
        GOTO 962
990     I=(IYMAX-IYMIN)*I/99+IYMIN
        IYMAX=(IYMAX-IYMIN)*J/99+IYMIN
        IYMIN=I

545     JTEMP=ITTOUR(26)

C   * * *  SET TIME SCALE LIMITS   * * *
        ITNEG=IPSR*IPSP+IHEAT
        ITPOS=ISR*ISP
        ITMINS=-ITNEG/60
        ITMAXS=ITPOS/60
        ISTART=ITNEG+ITMINS*60
        ITOTAL=ITNEG+ITPOS

        TYPE 1010
1010    FORMAT('DO YOU WANT TO EXPAND TIME SCALE?')
        ACCEPT 1015,I
1015    FORMAT(A)
        IF(I.NE.'Y') GOTO 1100
1020    TYPE 1025,ITMINS,ITMAXS
1025    FORMAT('ENTER LOWER TIME BOUNDARY - - FROM ',I3,' TO ',I3)
        ACCEPT 1027,I
1027    FORMAT(I3)
        IF(I.GE.ITMAXS.OR.I.LT.ITMINS) GOTO 1020
1029    TYPE 1030,I,ITMAXS
1030    FORMAT('ENTER UPPER TIME BOUNDARY - - FROM ',I3,' TO ',I3)
        ACCEPT 1032,J
1032    FORMAT(I3)
        IF(J.GT.ITMAXS.OR.J.LE.I) GOTO 1029
        ITMAXS=J
        ITMINS=I
        ISTART=0
        ITOTAL=(ITMAXS-ITMINS)*60
1100    JTEMP=ITTOUR(25)

C   * * *  LABEL CURVE   * * *
```

```
      JTEMP=ITTOUR(29)
      IX=0
      IY=779
      CALL GPOINT
      JTEMP=ITTOUR(13)
      TYPE 810,ITEXT
810   FORMAT(30X,20A2)
      TYPE 820,IFILEA(3),IFILEA(4),IFILEA(5),ICHAN
820   FORMAT(30X,'FILENAME:   ',3A2,'    CHANNEL:   ',I)
      TYPE 830,IPROBE,IHEAT/60,IHEAT-IHEAT/60*60
830   FORMAT(30X,'PROBE:   ',I2,'     HEAT PULSE:   ',I2,':',I2)
      CALL CVTTIM(ISTIM,KHRS,KMIN,KSEC,KTICKS)
      TYPE 840,KHRS,KMIN,KSEC,KDATE
840   FORMAT(30X,'SAMPLE TIME- ',I2,':',I2,':',I2,'   DATE- ',
     1        I2,'/',I2,'/',I2)
      JTEMP=ITTOUR(28)


C   * * *  DRAW AXIS   * * *
      JTEMP=ITTOUR(29)
      IX=IXMIN
      IY=IYMAX
      CALL GPOINT
      IX=IXMIN
      IY=IYMIN
      CALL GPOINT
      IX=IXMAX
      IY=IYMIN
      CALL GPOINT
      JTEMP=ITTOUR(28)

C   * * *  LABEL AXIS   * * *
      B=(IYMAX-IYMIN)/10.5
      YMID=(IYMAX-IYMIN)/2.0+IYMIN
      JTEMP=ITTOUR(31)
      TYPE 540
540   FORMAT()
      DO 580 I=-5,5
      IY=INT(YMID+I*B) + 11
      IX=IXMIN-35
      JTEMP=ITTOUR(29)
      CALL GPOINT
      JTEMP=ITTOUR(31)
      TYPE 550,I
550   FORMAT(I2)
      JTEMP=ITTOUR(29)
      IY=IY-11
      IX=IXMIN-5
      CALL GPOINT
      IX=IXMIN
      CALL GPOINT
      JTEMP=ITTOUR(28)
580   CONTINUE

      D=FLOAT(IXMAX-IXMIN)/ITOTAL
      DO 650 I=ITMINS,ITMAXS
      IX=INT((ISTART+(I-ITMINS)*60)*D)+IXMIN
      IY=IYMIN
      JTEMP=ITTOUR(29)
      CALL GPOINT
      IY=IY-5
      CALL GPOINT
      JTEMP=ITTOUR(28)
      JTEMP=ITTOUR(29)
```

```
        IX=IX-15
        CALL GPOINT
        JTEMP=ITTOUR(31)
        TYPE 640,I
640     FORMAT(I2)
        JTEMP=ITTOUR(28)
650     CONTINUE

C  * * *  PLOT CURVE  * * *
        ICOUNT=0
        ITLOW=ITMINS*60+ITNEG-ISTART
        ITHIGH=ITMAXS*60+ITNEG-ISTART
        D=FLOAT(IXMAX-IXMIN)/(ITHIGH-ITLOW)
        B=FLOAT(IYMAX-IYMIN)/"10000
        DO 720 I=1,IPSR+ISR
        IT=IPSP*I
        IF(I.GT.IPSR) IT=ITNEG+ISP*(I-IPSR)
        IF(IT.LT.ITLOW.OR.IT.GT.ITHIGH) GOTO 720
        IF(IT.LT.ITNEG.AND.IT.GT.IPSP*IPSR) GOTO 720
        IX=IXMIN+INT((IT-ITLOW)*D)
        ICOUNT=ICOUNT+1
        IF(ICOUNT.NE.15)GOTO 705
        ICOUNT=0
        JTEMP=ITTOUR(31)
        TYPE 703
703     FORMAT()
        JTEMP=ITTOUR(28)
705     IY=IDATA(I)
        IF(IY.GT."3777) IY=IY-"10000
        IY=INT(YMID+B*IY)
        CALL GPOINT
720     CONTINUE

        JTEMP=ITTOUR(31)
        JTEMP=ITTOUR(24)
        HAZA(3)=37
        HAZA(4)=72
        CALL PRINT(HAZA)



C  * * * RAP THINGS UP  * * *
290     TYPE 295
295     FORMAT('ENTER "E" TO EXIT OR "D" FOR MORE DATA ',$)
        ACCEPT 300,I
300     FORMAT(A)
        JTEMP=ITTOUR(28)
        JTEMP=ITTOUR(31)
        JTEMP=ITTOUR(25)
        JTEMP=ITTOUR(24)
        JTEMP=ITTOUR(26)
        IF(I.EQ.'E') GOTO 310
        TYPE 303
303     FORMAT('DO YOU WANT TO REMAIN WITH SAME DATA ?')
        ACCEPT 307,I
307     FORMAT(A)
        IF(I.EQ.'Y') GOTO 65
        GOTO 10
310     RETURN
        END
```

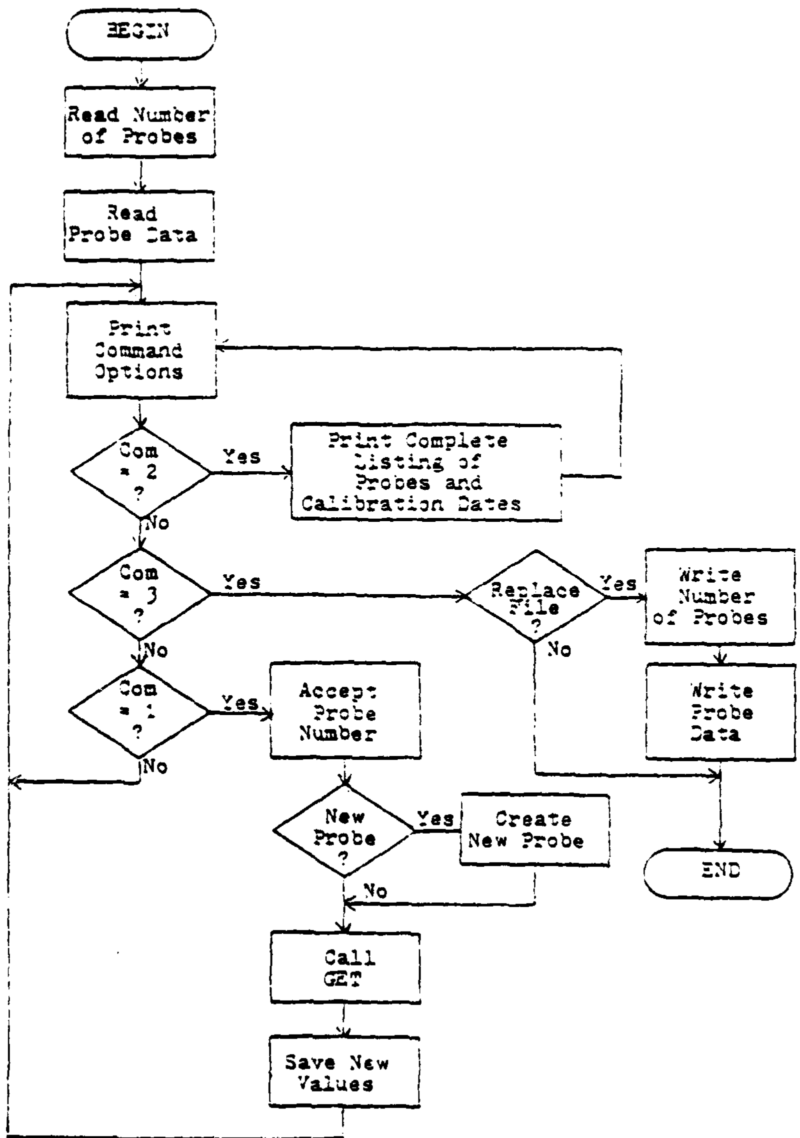


Figure 21. Flowchart of PROBEC.FOR

```
C                PROBEC FOR
C
C       THIS PROGRAM MODIFIES THE FILE PROBE.DAT
C       AND USES SUBROUTINE GET TO ACCEPT NEW CALIBRATIONS

        DIMENSION IDATA(1500)
        COMMON IDESIG,IMONTH,IDAY,IYEAR,ATEM,BTEM,RB0TEM,BETAT
        DIMENSION JDATA(12)
        EQUIVALENCE (ATEM,JDATA(5)),(BTEM,JDATA(7))
        EQUIVALENCE (RB0TEM,JDATA(9)),(BETAT,JDATA(11))
        EQUIVALENCE (IDESIG,JDATA(1)),(IMONTH,JDATA(2))
        EQUIVALENCE (IDAY,JDATA(3)),(IYEAR,JDATA(4))
        CALL ASSIGN(3,'FO0:PROBE.DAT',13)
        READ (3) JNUM
        INUM=JNUM*12
        READ (3) (IDATA(K),K=1,INUM)
        CALL CLOSE(3)
5       TYPE 6
6       FORMAT( )
        TYPE 2
2       FORMAT( 'ENTER   "1"   TO ADD OR MODIFY A PROBE )
        TYPE 3
3       FORMAT( 'ENTER   "2"   TO LIST ALL PROBES')
        TYPE 4
4       FORMAT( 'ENTER   "3"   TO EXIT')
7       ACCEPT 8,ICOM
8       FORMAT( I )
        IF (ICOM.EQ.2) GOTO 200
        IF (ICOM.EQ.3) GOTO 300
        IF (ICOM.NE.1) GOTO 5
        TYPE 6
        TYPE 25


C  * * *  ADD OR MODIFY A PROBE  * * *
25      FORMAT( 'ENTER PROBE NUMBER' )
        ACCEPT 30,IDESIG
30      FORMAT( I3 )
        TYPE 6
        INDEX=1
40      IF (IDESIG.EQ.IDATA(INDEX)) GOTO 150
        INDEX=INDEX+12
        IF (INDEX.LT.INUM) GOTO 40
        TYPE 100
100     FORMAT( 'NEW PROBE NUMBER' )
        TYPE 110
110     FORMAT( 'ENTER "Y" TO START A NEW PROBE WITH THAT',
     1         ' DESIGNATION' )
        ACCEPT 112,ICOM
112     FORMAT( A )
        TYPE 6
        IF (ICOM.NE.'Y') GOTO 5
        CALL IDATE(IMONTH,IDAY,IYEAR)
        ATEM=0
        BTEM=0
        RB0TEM=0
        BETAT=0
        JNUM=JNUM+1
        INDEX=INUM+1
        INUM=INUM+12
        CALL GET
120     DO 122 K=1,12
        IDATA(INDEX)=JDATA(K)
        INDEX=INDEX+1
122     CONTINUE
```

```
      GOTO 5
150   TYPE 155
155   FORMAT( 'PROBE CURRENTLY HAS THESE VALUES' )
      DO 160 K=1,12
      JDATA(K)=IDATA(INDEX)
      INDEX=INDEX+1
160   CONTINUE
      INDEX=INDEX-12
      CALL GET
      GOTO120
200   TYPE 6
      TYPE 202

C  * * *  LIST ALL PROBES IN FILE  * * *
202   FORMAT( 'THIS IS A COMPLETE LISTING OF PROBES AND DATES' )
      DO 210 K=1,JNUM
      INDEX=K*12-11
      DO 205 J=1,4
      JDATA(J)=IDATA(INDEX)
      INDEX=INDEX+1
205   CONTINUE
      TYPE 207,IDESIG,IMONTH,IDAY,IYEAR
207   FORMAT( I3,5X,I2,'/',I2,'/',I2)
210   CONTINUE
      GOTO 5

C  * * *  CHECK IF REPLACEMENT DESIRED  * * *
300   TYPE 310
310   FORMAT( 'LAST CHANCE TO REMAIN WITH EXISTING PROBE FILE' )
      TYPE 315
315   FORMAT( 'ENTER  "1" TO REPLACE OLD PROBE FILE' )
      ACCEPT 320,ICOM
320   FORMAT( I )
      IF (ICOM.NE.1) GOTO 400
      CALL ASSIGN(3,'F00:PROBE.DAT',13)
      WRITE(3)JNUM
      WRITE(3)IDATA
      CALL CLOSE(3)
      TYPE 325
325   FORMAT( 'NEW FILE CREATED - - OLD FILE DESTROYED' )
      GOTO 490
400   TYPE 410
410   FORMAT( 'NO CHANGES MADE TO EXISTING PROBE FILE' )
500   FORMAT( ' STILL UNDER CONSTRUCTION ' )
490   STOP'      THAT IS ALL FOR THIS MESS    '
      END

C               CREATE  PROBE.DAT

C     THIS PROGRAM CREATES AN EMPTY FILE
C     TO BE USED BY PROBEC.FOR

      DIMENSION JDATA(12)
      CALL ASSIGN(3,'F00:PROBE.DAT',13)
      WRITE(3)1
      DO 5 K=1,12
      JDATA(K)=0
5     CONTINUE
      WRITE(3)JDATA
      CALL CLOSE(3)
      STOP'FILE CREATED'
      END
```

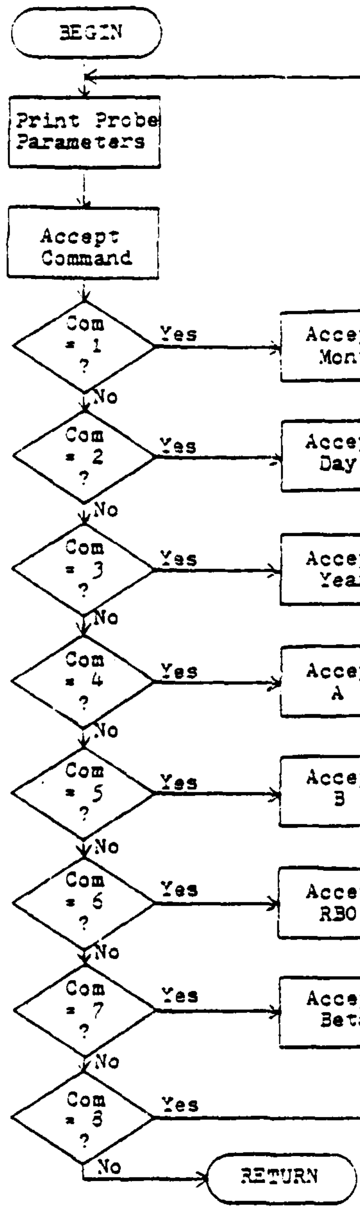


Figure 22. Flowchart of Subroutine GET

```
      SUBROUTINE GET
C     THIS SUBROUTINE ALLOWS THE USER TO CHANGE THE
C     CALIBRATION DATA FOR A GIVEN PROBE
      COMMON IDESIG,IMONTH,IDAY,IYEAR,ATEM,BTEM,RB0TEM,BETAT
3     TYPE 5,IDESIG
5     FORMAT( 'PROBE NUMBER =     ',I3,'    COMMAND')
      TYPE 10,IMONTH
10    FORMAT( 'MONTH        =     ',I3,'      (1)')
      TYPE 15,IDAY
15    FORMAT( 'DAY          =     ',I3,'      (2)')
      TYPE 20,IYEAR
20    FORMAT( 'YEAR         =     ',I3,'      (3)')
      TYPE 25,ATEM
25    FORMAT( 'A            = ',F8.4,'   (4)')
      TYPE 30,BTEM
30    FORMAT( 'B            = ',F8.4,'   (5)')
      TYPE 35,RB0TEM
35    FORMAT( 'RB0          = ',F8.2,'   (6)')
      TYPE 40,BETAT
40    FORMAT( 'BETA         = ',F8.3,'   (7)')
      TYPE 45
45    FORMAT( 'TO EXIT TYPE                (8)')
      ACCEPT 100,ICOM
100   FORMAT( I )
      IF (ICOM.NE.1) GOTO 110
      TYPE 105
105   FORMAT( 'ENTER MONTH')
      ACCEPT 106,IMONTH
106   FORMAT( I2 )
110   IF (ICOM.NE.2) GOTO 120
      TYPE 115
115   FORMAT( 'ENTER DAY' )
      ACCEPT 106,IDAY
120   IF (ICOM.NE.3) GOTO 130
      TYPE 125
125   FORMAT( 'ENTER YEAR' )
      ACCEPT 106,IYEAR
130   IF (ICOM.NE.4) GOTO 140
      TYPE 135
135   FORMAT( 'ENTER A' )
      ACCEPT 136,ATEM
136   FORMAT( F8.4 )
140   IF (ICOM.NE.5) GOTO 150
      TYPE 145
145   FORMAT( 'ENTER B' )
      ACCEPT 136,BTEM
150   IF (ICOM.NE.6) GOTO 160
      TYPE 155
155   FORMAT( 'ENTER RB0' )
      ACCEPT 136,RB0TEM
160   IF (ICOM.NE.7) GOTO 170
      TYPE 165
165   FORMAT( 'ENTER BETA' )
      ACCEPT 136,BETAT
170   IF (ICOM.NE.8) GOTO 3
      RETURN
      END
```

REFERENCES

1. Chen, M. M. and K. R. Holmes. "The thermal pulse-decay method for simultaneous measurement of thermal conductivity and local blood perfusion rate of living tissues." 1980 Advances in Bioengineering, Am. Soc. Mech. Engr., 1980, pp. 113-115.
2. Aukland, K. "Methods for measuring renal blood flow: total flow and regional distribution." Ann. Rev. Physiol. 42:543-555, 1980.
3. Holmes, K. R. and M. M. Chen. "In vivo tissue thermal conductivity and local blood perfusion measured with a heat pulse-decay method." Physiologist 23:183, 1980.
4. Holmes, K. R. and M. M. Chen. Local thermal conductivity of Para-7 fibrosarcoma in hamster. 1979 Advances in Bioengineering, Am Soc. Mech. Engr., 1979, 147-149.
5. Digital Equipment Corp. Microcomputer Processors (1978).
6. Digital Equipment Corp. Memories and Peripherals (1979).
7. Adac Corporation. Model 1030 Analog to Digital Converter Manual (1981).
8. Lear Sigler, Inc. ADM 5 Dumb Terminal Video Display Unit Users Reference Manual (1981).
9. Digital Engineering, Inc. User's Manual RG-512 Retro-graphics Card for the ADM-3A Computer Terminal (1981).
10. Digital Engineering, Inc. GP-100 Graphx Printer User's Manual (1981).
11. Digital Equipment Corp. RT-11 System Reference Manual (1979).